50 Years of CORDIC: Algorithms, Architectures, and Applications

Pramod K. Meher, *Senior Member, IEEE*, Javier Valls, *Member, IEEE*, Tso-Bing Juang, *Member, IEEE*, K. Sridharan, *Senior Member, IEEE*, and Koushik Maharatna, *Member, IEEE*

***Abstract*—Year 2009 marks the completion of 50 years of the invention of CORDIC (COordinate Rotation DIgital Computer) by Jack E. Volder. The beauty of CORDIC lies in the fact that by simple shift-add operations, it can perform several computing tasks such as the calculation of trigonometric, hyperbolic and logarithmic functions, real and complex multiplications, division, square-root, solution of linear systems, eigenvalue estimation, singular value decomposition, QR factorization and many others. As a consequence, CORDIC has been utilized for applications in diverse areas such as signal and image processing, communication systems, robotics and 3-D graphics apart from general scientific and technical computation. In this article, we present a brief overview of the key developments in the CORDIC algorithms and architectures along with their potential and upcoming applications.**

***Index Terms*—Arithmetic circuits, CORDIC, CORDIC algorithms, digital signal processing chip, VLSI.**

I. Introduction

COORDINATE Rotation DIgital Computer is abbreviated as CORDIC. The key concept of CORDIC arithmetic is based on the simple and ancient principles of two-dimensional geometry. But the iterative formulation of a computational algorithm for its implementation was first described in 1959 by Jack E. Volder [1], [2] for the computation of trigonometric functions, multiplication and division. This year therefore marks the completion of 50 years of the CORDIC algorithm. Not only a wide variety of applications of CORDIC have emerged in the last 50 years, but also a lot of progress has been made in the area of algorithm design and development of architectures for high-performance and low-cost hardware solutions of those applications. CORDIC-based computing received increased attention in 1971, when John Walther [3], [4] showed that, by varying a few simple parameters, it could be used as a single algorithm for unified implementation of a wide range of elementary transcendental functions involving logarithms, exponentials, and square roots along with those suggested by Volder [1]. During the same time, Cochran [5] benchmarked various algorithms, and showed that CORDIC technique is a better choice for scientific calculator applications.

The popularity of CORDIC was very much enhanced thereafter primarily due to its potential for efficient and low-cost implementation of a large class of applications which include: the generation of trigonometric, logarithmic and transcendental elementary functions; complex number multiplication, eigenvalue computation, matrix inversion, solution of linear systems and singular value decomposition (SVD) for signal processing, image processing, and general scientific computation. Some other popular and upcoming applications are:

1) direct frequency synthesis, digital modulation and coding for speech/music synthesis and communication;
2) direct and inverse kinematics computation for robot manipulation;
3) planar and three-dimensional vector rotation for graphics and animation.

Although CORDIC may not be the fastest technique to perform these operations, it is attractive due to the simplicity of its hardware implementation, since the same iterative algorithm could be used for all these applications using the basic shift-add operations of the form $a \pm b.2^{-i}$.

Keeping the requirements and constraints of different application environments in view, the development of CORDIC algorithm and architecture has taken place for achieving high throughput rate and reduction of hardware-complexity as well as the latency of implementation. Some of the typical approaches for reduced-complexity implementation are focussed on minimization of the complexity of scaling operation and the complexity of barrel-shifter in the CORDIC engine. Latency of implementation is an inherent drawback of the conventional CORDIC algorithm. Angle recoding schemes, mixed-grain rotation and higher radix CORDIC have been developed for reduced latency realization. Parallel and pipelined CORDIC have been suggested for high-throughput computation. The objective of this article is not to present a detailed survey of the developments of algorithms, architectures and applications of CORDIC, which would require a few doctoral and masters level dissertations. Rather we aim at providing the key developments in algorithms and architectures along with an overview

Manuscript received August 22, 2008; revised November 26, 2008 and April 10, 2009. First published June 19, 2009; current version published September 02, 2009. This paper was recommended by Associate Editor V. Öwall.

P. K. Meher is with the Department of Communication Systems, Institute for Infocomm Research, Singapore 138632 (e-mail: pkmeher@i2r.a-star.edu.sg).

J. Valls is with Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universidad Politécnica de Valencia, 46730 Grao de Gandia, Spain (e-mail: jvalls@eln.upv.es).

T.-B. Juang is with the Department of Computer Science and Information Engineering, National Pingtung Institute of Commerce, Pingtung City, Taiwan 900 (e-mail: tsobing@npic.edu.tw).

K. Sridharan is with the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai 600036, India (e-mail: sridhara@iitm.ac.in).

K. Maharatna is with the School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, U.K. (e-mail: km3@ecs.soton.ac.uk).

Digital Object Identifier 10.1109/TCSI.2009.2025803

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

of the major application areas and upcoming applications. We shall however discuss here the basic principles of CORDIC operations for the benefit of general readers.

The remainder of this paper is organized as follows. In Section II, we discuss the principles of CORDIC operation, covering the elementary ideas from coordinate transformation to rotation mode and vectoring mode operations followed by design of the basic CORDIC cell and multidimensional CORDIC. The key developments in CORDIC algorithms and architectures are discussed in Section III, which covers the algorithms and architectures pertaining to higher-radix CORDIC, angle recording, coarse-fine hybrid micro rotations, redundant number representation, differential CORDIC, and pipeline implementation. In Section IV, we discuss the scaling and accuracy aspects including the scaling techniques, scaling-free CORDIC, quantization and area-delay-accuracy trade-off. The applications of CORDIC to scientific computations, signal processing, communications, robotics and graphics are discussed briefly in Section V. The conclusion along with future research directions are discussed in Section VI.

II. Basic CORDIC Techniques

In this Section, we discuss the basic principle underlying the CORDIC-based computation, and present its iterative algorithm for different operating modes and planar coordinate systems. At the end of this section, we discuss the extension of two-dimensional rotation to multidimensional formulation.

A. *The CORDIC Algorithm*

As shown in Fig. 1, the rotation of a two-dimensional vector $\mathbf{p}_0 = [x_0\ y_0]$ through an angle θ, to obtain a rotated vector $\mathbf{p}_n = [x_n\ y_n]$ could be performed by the matrix product $\mathbf{p}_n = \mathbf{R}\mathbf{p}_0$, where $\mathbf{R}$ is the rotation matrix:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{1}$$

By factoring out the cosine term in (1), the rotation matrix $\mathbf{R}$ can be rewritten as

$$\mathbf{R} = [(1+\tan^2\theta)^{-1/2}] \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \tag{2}$$

and can be interpreted as a product of a scale-factor $K = [(1+\tan^2\theta)^{-1/2}]$ with a pseudorotation matrix $\mathbf{R_c}$, given by

$$\mathbf{R_c} = \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix}. \tag{3}$$

The pseudorotation operation rotates the vector $\mathbf{p}_0$ by an angle θ and changes its magnitude by a factor $K = \cos\theta$, to produce a pseudo-rotated vector $\mathbf{p}'_n = \mathbf{R_c}\mathbf{p}_0$.

To achieve simplicity of hardware realization of the rotation, the key ideas used in CORDIC arithmetic are to (i) decompose the rotations into a sequence of elementary rotations through predefined angles that could be implemented with minimum hardware cost; and (ii) to avoid scaling, that might involve arithmetic operation, such as square-root and division. The second idea is based on the fact the scale-factor contains only the magnitude information but no information about the angle of rotation.

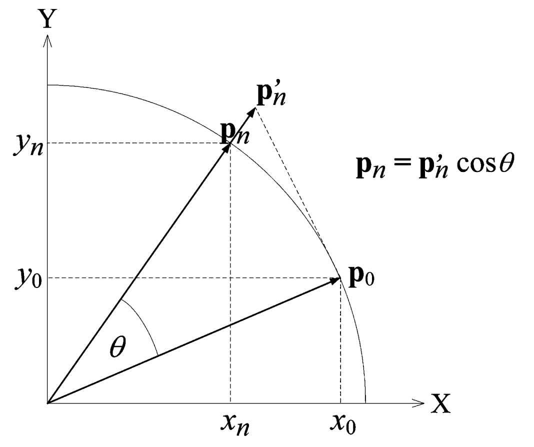

Fig. 1. Rotation of vector on a two-dimensional plane.

1) Iterative Decomposition of Angle of Rotation: The CORDIC algorithm performs the rotation iteratively by breaking down the angle of rotation into a set of small pre-defined angles[1], $\alpha_i = \arctan(2^{-i})$, so that $\tan\alpha_i = 2^{-i}$ could be implemented in hardware by shifting through i bit locations. Instead of performing the rotation directly through an angle θ, CORDIC performs it by a certain number of microrotations through angle α_i, where

$$\theta = \sum_{i=0}^{n-1} \sigma_i \alpha_i, \quad \text{and } \sigma_i = \pm 1 \tag{4}$$

that satisfies the CORDIC *convergence theorem* [3]: $\alpha_i - \Sigma_{j=i+1}^{n-1}\alpha_j < \alpha_{n-1}, \forall i, i = 0, 1, \ldots, n-2$. But, the decomposition according to (4) could be used only for $-1.74329 \leq \theta \leq 1.74329$ (called the "convergence range") since $\Sigma_{i=0}^{\infty}(\alpha_i) = 1.743286\ldots$. Therefore, the angular decomposition of (4) is applicable for angles in the first and fourth quadrants. To obtain on-the-fly decomposition of angles into the discrete base α_i, one may otherwise use the nonrestoring decomposition [6]

$$\omega_0 = 0 \quad \text{and } \omega_{i+1} = \omega_i - \sigma_i \cdot \alpha_i \tag{5}$$

with $\sigma_i = 1$ if $\omega_i \geq 0$ and $\sigma_i = -1$ otherwise, where the rotation matrix for the ith iteration corresponding to the selected angle α_i is given by

$$\mathbf{R}(i) = K_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \tag{6}$$

$K_i = 1/\sqrt{(1+2^{-2i})}$ being the scale-factor, and the pseudorotation matrix

$$\mathbf{R_c}(i) = \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix}. \tag{7}$$

Note that the pseudo-rotation matrix $\mathbf{R_c}(i)$ for the ith iteration alters the magnitude of the rotated vector by a scale-factor $K_i = 1/\sqrt{(1+2^{-2i})}$ during the ith microrotation, which is independent of the value of σ_i (direction of microrotation) used in the angle decomposition.

[1]All angles are measured in radian unless otherwise stated.

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

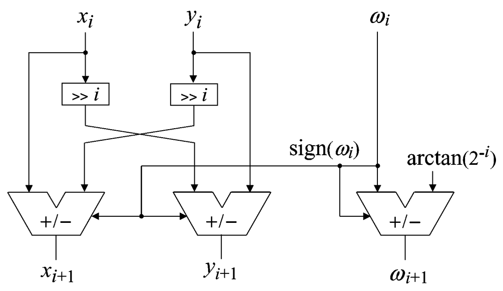

Fig. 2. Hardware implementation of a CORDIC iteration.

2) Avoidance of Scaling: The other simplification performed by the Volder's algorithm [1] is to remove the scale-factor $K_i = 1/\sqrt{(1+2^{-2i})}$ from (6). The removal of scaling from the iterative microrotations leads to a pseudo-rotated vector $\mathbf{p}'_n = \mathbf{R_c p}_0$ instead of the desired rotated vector $\mathbf{p}_n = K\mathbf{R_c p}_0$, where the scale-factor K is given by

$$K = \prod_{i=0}^{n} K_i = \prod_{i=0}^{n} 1/\sqrt{(1+2^{-2i})}. \tag{8}$$

Since the scale-factor of microrotations does not depend on the direction of microrotations and decreases monotonically, the final scale-factor K converges to $\sim$1.6467605. Therefore, instead of scaling during each microrotation, the magnitude of final output could be scaled by K. Therefore, the basic CORDIC iterations are obtained by applying the pseudo-rotation of the vector to have, $\mathbf{p}'_{i+1} = \mathbf{R_c}(i)\mathbf{p}_i$, together with the nonrestoring decomposition of the selected angles α_i, as follows:

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i \cdot 2^{-i} \cdot y_i \\ y_{i+1} &= y_i + \sigma_i \cdot 2^{-i} \cdot x_i \\ \omega_{i+1} &= \omega_i - \sigma_i \cdot \alpha_i \end{aligned} \tag{9}$$

CORDIC iterations of (9) could be used in two operating modes, namely the *rotation mode* (RM) and the *vectoring mode* (VM), which differ basically on how the directions of the microrotations are chosen. In the rotation mode, a vector $\mathbf{p}_0$ is rotated by an angle θ to obtain a new vector $\mathbf{p'}_n$. In this mode, the direction of each microrotation σ_i is determined by the sign of ω_i: if sign of ω_i is positive, then $\sigma_i = 1$ otherwise $\sigma_i = -1$. In the vectoring mode, the vector $\mathbf{p}_0$ is rotated towards the x-axis so that the y-component approaches zero. The sum of all angles of microrotations (output angle ω_n) is equal to the angle of rotation of vector $\mathbf{p}_0$, while output x'_n corresponds to its magnitude. In this operating mode, the decision about the direction of the microrotation depends on the sign of y_i: if it is positive then $\sigma_i = -1$ otherwise $\sigma_i = 1$. CORDIC iterations are easily implemented in both software and hardware. Fig. 2 shows the basic hardware stage for a single CORDIC iteration. After each iteration the number of shifts is incremented by a pair of barrel-shifters. To have an n-bit output precision, $(n+1)$ CORDIC iterations are needed. Note that it could be implemented by a simple selection operation in serial architectures like the one proposed in the original work, or in fully parallel CORDIC architectures the shift operations could be hardwired, where no barrel-shifters are involved.

Finally, to overcome the problem of the limited convergence range and, then to extend the CORDIC rotations to the complete range of $\pm\pi$, an extra iteration is required to be performed. This new iteration is shown in (10) which is required as an initial rotation through $\pm\pi/2$.

$$\begin{aligned} x_0 &= -\sigma_{-i} \cdot y_{-i} \\ y_0 &= \sigma_{-i} \cdot x_{-i} \\ \omega_0 &= \omega_{-i} - \sigma_{-i} \cdot \alpha_{-i} \quad \text{where } \alpha_{-i} = \pi/2. \end{aligned} \tag{10}$$

TABLE I
GENERALIZED CORDIC ALGORITHM

m	rotation mode	vectoring mode
0	$x_n = K(x_o \cos\omega_0 - y_o \sin\omega_0)$	$x_n = K\sqrt{x_o^2 + y_0^2}$
	$y_n = K(y_o \cos\omega_0 + y_o \sin\omega_0)$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + \tan^{-1}(y_0/x_0)$
1	$x_n = x_o$	$x_n = x_o$
	$y_n = y_o + x_o\omega_0$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + (y_0/x_0)$
-1	$x_n = K_h(x_o \cosh\omega_0 - y_o \sinh\omega_0)$	$x_n = K_h\sqrt{x_o^2 - y_0^2}$
	$y_n = K_h(y_o \cosh\omega_0 + y_o \sinh\omega_0)$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + \tanh^{-1}(y_0/x_0)$

B. Generalization of the CORDIC Algorithm

In 1971, Walther found how CORDIC iterations could be modified to compute hyperbolic functions [3] and reformulated the CORDIC algorithm in to a generalized and unified form which is suitable to perform rotations in circular, hyperbolic and linear coordinate systems. The unified formulation includes a new variable m, which is assigned different values for different coordinate systems. The generalized CORDIC is formulated as follows:

$$\begin{aligned} x_{i+1} &= x_i - m\sigma_i \cdot 2^{-i} \cdot y_i \\ y_{i+1} &= y_i + \sigma_i \cdot 2^{-i} \cdot x_i \\ \omega_{i+1} &= \omega_i - \sigma_i \cdot \alpha_i \end{aligned} \tag{11}$$

where

$$\sigma_i = \begin{cases} \text{sign}(\omega_i), & \text{for rotation mode} \\ -\text{sign}(y_i), & \text{for vectoring mode} \end{cases}$$

For $m = 1, 0$ or -1, and $\alpha_i = \tan^{-1}(2^{-i}), 2^{-i}$ or $\tanh^{-1}(2^{-i})$, the algorithm given by (11) works in circular, linear or hyperbolic coordinate systems, respectively. Table I summarizes the operations that can be performed in rotation and vectoring modes[2] in each of these coordinate systems. The convergence range of linear and hyperbolic CORDIC are obtained, as in the case of circular coordinate, by the sum of all α_i given by $C = \Sigma_{i=0}^{\infty}\alpha_i$. The hyperbolic CORDIC requires to execute iterations for $i = 4, 13, 40, \ldots$ twice to ensure convergence. Consequently, these repetitions must be considered while computing the scale-factor $K_h = \Pi(1+2^{-2i})^{-1/2}$, which converges to 0.8281.

[2]In the *rotation mode*, the components of a vector resulting due to rotation of a vector through a given angle are derived, while in the *vectoring mode* the magnitude as well as the phase angle of a vector are estimated from the component values. The rotation and vectoring modes are also known as the *vector rotation mode* and the *angle accumulation mode*, respectively.

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

C. Multidimensional CORDIC

The CORDIC algorithm was extended to higher dimensions using simple Householder reflection [7]. The Householder reflection matrix is defined as

$$\mathbf{H}_m = \mathbf{I}_m - 2.\frac{\mathbf{u}\mathbf{u}^{\mathrm{T}}}{\mathbf{u}\mathbf{u}} \tag{12}$$

where $\mathbf{u}$ is an m-dimensional vector and $\mathbf{I}_m$ is the $m \times m$ identity matrix. The product $(\mathbf{H}_m\mathbf{v})$ reflects the m-dimensional vector $\mathbf{v}$ with respect to the hyperplane with normal $\mathbf{u}$ that passes through the origin. Basically, the Householder-based CORDIC performs the vectoring operation of an m-dimensional vector to one of the axes.

For the sake of clarity, we consider here the case of 3-D vector $P_0 = [x_0\ y_0\ z_0]$ projected on to the x-axis in the Euclidean space. The rotation matrix for 3-D case, corresponding to the ith iteration, $\mathbf{R}_{H3}(i)$, is given by the product of two simple Householder reflections as

$$\mathbf{R}_{H3}(i) = \left[\mathbf{I}_3 - 2.\frac{\mathbf{e}_1\mathbf{e}_1^{\mathrm{T}}}{\mathbf{e}_1^{\mathrm{T}}\mathbf{e}_1}\right] \cdot \left[\mathbf{I}_3 - 2.\frac{\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}}{\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_i}\right] \tag{13}$$

where $\mathbf{e}_1 = [1\ 0\ 0]^{\mathrm{T}}$, and $\mathbf{u}_i = [1\ \sigma_{yi}t_i\ \sigma_{zi}t_i]^{\mathrm{T}}$ with $t_i = \tan\alpha_i = 2^{-i}$, and $\sigma_{yi} = \mathrm{sign}\ (x_iy_i)$ and $\sigma_{zi} = \mathrm{sign}\ (x_iz_i)$ being the directions of microrotations.

One can write the ith rotation matrix in terms of the pseudo-rotation matrix as $\mathbf{R}_{H3}(i) = K_{Hi}\mathbf{R}_{HC3}(i)$, where $K_{Hi} = 1/\sqrt{(1+2^{-2i+1})}$ is the scale-factor and $\mathbf{R}_{HC3}(i)$ is the pseudo-rotation matrix which could be expressed as function of the shifting and decision variables as

$$\begin{bmatrix} 1 & \sigma_{yi}2^{-i+1} & \sigma_{zi}2^{-i+1} \\ -\sigma_{yi}2^{-i+1} & 1-2^{-2i+1} & -\sigma_{yi}\sigma_{zi}2^{-2i+1} \\ -\sigma_{zi}2^{-i+1} & -\sigma_{zi}\sigma_{yi}2^{-2i+1} & 1-2^{-2i+1} \end{bmatrix}. \tag{14}$$

Therefore, the ith iteration of 3-D Housholder CORDIC rotation results $\mathbf{p}_{i+1} = \mathbf{R}_{HC3}(i)\mathbf{p}_i$, and, the vector is projected to x-axis, such that after n iterations x_n gives the length of the vector scaled by $\Pi_{i=0}^{n}(K_{Hi})$ with $(n-1)$ bit precision [8].

III. Advanced CORDIC Algorithms and Architectures

CORDIC computation is inherently sequential due to two main bottlenecks: 1) the micro-rotation for any iteration is performed on the intermediate vector computed by the previous iteration and 2) the $(i+1)$th iteration could be started only after the completion of the ith iteration, since the value of σ_{i+1} which is required to start the $(i+1)$th iteration could be known only after the completion of the ith iteration. To alleviate the second bottleneck some attempts have been made for evaluation of σ_i values corresponding to small micro-rotation angles [9], [10]. However, the CORDIC iterations could not still be performed in parallel due to the first bottleneck. A partial parallelization has been realized in [11] by combining a pair of conventional CORDIC iterations into a single merged iteration which provides better area-delay efficiency. But the accuracy is slightly affected by such merging and cannot be extended to a higher number of conventional CORDIC iterations since the induced error becomes unacceptable [11]. Parallel realization of CORDIC iterations to handle the first bottleneck by direct unfolding of micro-rotation is possible, but that would result in increase in computational complexity and the advantage of simplicity of CORDIC algorithm gets degraded [12], [13]. Although no popular architectures are known to us for fully parallel implementation of CORDIC, different forms of pipelined implementation of CORDIC have however been proposed for improving the computational throughput [14].

Since the CORDIC algorithm exhibits linear-rate convergence, it requires $(n+1)$ iterations to have n-bit precision of the output. Overall latency of the computation thus amounts to product of the word-length and the CORDIC iteration period. The speed of CORDIC operations is therefore constrained either by the precision requirement (iteration count) or the duration of the clock period. The duration of clock period on the other hand mainly depends on the large carry propagation time for the addition/subtraction during each micro-rotation. It is a straight-forward choice to use fast adders for reducing the iteration period at the expense of large silicon area. Use of carry-save adder is a good option to reduce the iteration period and overall latency [15]. Timmermann and others have suggested a method of truncation of CORDIC algorithm after $(n+1)/2$ iterations (for n-bit precision), where the last iteration performs a single rotation for implementing the remaining angle. It lowers the the latency time but involves one multiplication or division, respectively, in the rotation or vectoring mode [9].

To handle latency bottlenecks, various techniques have been developed and reported in the literature. Most of the well known algorithms could be grouped under, high-radix CORDIC, the angle-recoding method, hybrid micro-rotation scheme, redundant CORDIC and differential CORDIC which we discuss briefly in the following subsections.

A. Higher Radix CORDIC Algorithm

The radix-4 CORDIC algorithm [16] is given by

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i \cdot 4^{-i} \cdot y_i \\ y_{i+1} &= y_i + \sigma_i \cdot 4^{-i} \cdot x_i \\ \omega_{i+1} &= \omega_i - \sigma_i \cdot \alpha_i \end{aligned} \tag{15}$$

where $\sigma_i \in \{-2,-1,0,1,2\}$ and the elementary angles $\alpha_i = \arctan(\sigma_i 4^{-i})$. The scale-factor for the ith iteration $K_i = 1/\sqrt{(1+\sigma_i^2 4^{-2i})}$. In order to preserve the norm of the vector the output of micro-rotations is required to be scaled by a factor

$$K = 1/\prod_i \sqrt{(1+\sigma_i^2 4^{-2i})}. \tag{16}$$

To have n-bit output precision, the radix-4 CORDIC algorithm requires $n/2$ micro-rotations, which is half that of radix-2 algorithm. However, it requires more computation time for each iteration and involves more hardware compared to the radix-2 CORDIC to select the value of σ_i out of five different possibilities. Moreover, the scale-factor, given by (16), also varies with the rotation angles since it depends on σ_i which could have

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

any of the five different values. Some techniques have therefore been suggested for scale-factor compensation through iterative shift-add operations [16], [17]. A high-radix CORDIC algorithm in vectoring mode is also suggested in [18], which can be used for reduced latency operation at the cost of larger size tables for storing the elementary angles and pre-scaling factors than the radix-2 and radix-4 implementation.

B. *Angle Recoding (AR) Methods*

The purpose of angle recoding (AR) is to reduce the number of CORDIC iterations by encoding the angle of rotation as a linear combination of a set of selected elementary angles of micro-rotations. AR methods are well-suited for many signal processing and image processing applications where the rotation angle is known *a priori*, such as when performing the discrete orthogonal transforms like discrete Fourier transform (DFT), the discrete cosine transform (DCT), etc.

1) Elementary-Angle-Set Recoding: In the conventional CORDIC, any given rotation angle is expressed as a linear combination of n values of elementary angles that belong to the set $S = \{(\sigma \cdot \arctan(2^{-r})) : \sigma \in \{-1, 1\}, r \in \{1, 2, \ldots, n-1\}\}$ in order to obtain an n-bit value as $\theta = \Sigma_{i=0}^{n-1}[\sigma_i \cdot \arctan(2^{-i})]$. However, in AR methods, this constraint is relaxed by adding zeros to the linear combination to obtain the desired angle using relatively fewer terms of the form $(\sigma \cdot \arctan 2^{-r})$ for $\sigma \in \{1, 0, -1\}$. The elementary-angle-set (EAS) used by AR scheme is given by $S_{\mathrm{EAS}} = \{(\sigma \arctan 2^{-r}) : \sigma \in \{-1, 0, 1\}, r \in \{1, 2, \ldots, n-1\}\}$. One of the simplest form of the angle recoding method based on the greedy algorithm proposed by Hu and Naganathan [19] tries to represent the remaining angle using the closest elementary angle $\pm \arctan 2^{-i}$. The angle recoding algorithm of [19] is briefly stated in Table II. Using this recoding scheme the total number of iterations could be reduced by at least 50% keeping the same n-bit accuracy unchanged. A similar method of angle recoding in vectoring mode called as the *backward angle recoding* is suggested in [20].

2) Extended Elementary-Angle-Set Recoding: Wu *et al.* [21] have suggested an AR scheme based on an extended elementary-angle-set (EEAS), that provides a more flexible way of decomposing the target rotation angle. In the EEAS approach, the set S_{EAS} of the elementary-angle set is extended further to $S_{\mathrm{EEAS}} = \{(\arctan(\sigma_1 \cdot 2^{-r_1} + \sigma_2 \cdot 2^{-r_2})) : \sigma_1, \sigma_2 \in \{-1, 0, 1\}$ and $r_1, r_2 \in \{1, 2, \ldots, n-1\}\}$. EEAS has better recoding efficiency in terms of the number of iterations and can yield better error performance than the AR scheme based on EAS. The pseudo-rotation for ith micro-rotations based on EEAS scheme is given by

$$\begin{aligned} x_{i+1} &= x_i - \left[\sigma_1(i) \cdot 2^{-r_1(i)} + \sigma_2(i) \cdot 2^{-r_2(i)}\right] y_i \\ y_{i+1} &= y_i + \left[\sigma_1(i) \cdot 2^{-r_1(i)} + \sigma_2(i) \cdot 2^{-r_2(i)}\right] x_i. \quad (17) \end{aligned}$$

The pseudo-rotated vector $[x_{R_m} \; y_{R_m}]$, obtained after R_m (the required number of micro-rotations) iterations, according to (17), needs to be scaled by a factor $K = \Pi(K_i)$, where $K_i = [1 + (\sigma_1(i) \cdot 2^{-r_1(i)} + \sigma_2(i) \cdot 2^{-r_2(i)})^2]^{-1/2}$ to produce the rotated vector. For reducing the scaling approximation and for a more flexible implementation of scaling, similar to the

TABLE II
ANGLE RECODING ALGORITHM

```
initialize: $\theta_0 = \theta$, $\sigma_i = 0$ for $0 \le i \le (n-1)$ and $k = 0$.
repeat until $|\theta_k| < \arctan(2^{-n+1})$ do:
1. choose $i_k, 0 \le i_k \le (n-1)$ such that
   $||\theta_k| - \arctan(2^{-i_k})| = \min_{0 \le i \le (n-1)} ||\theta_k| - \arctan(2^{-i})|$.
2. $\theta_{k+1} = \theta_k - \sigma_{i_k} \arctan(2^{-i_k})$, where $\sigma_{i_k} = \mathrm{sign}(\theta_k)$.
```

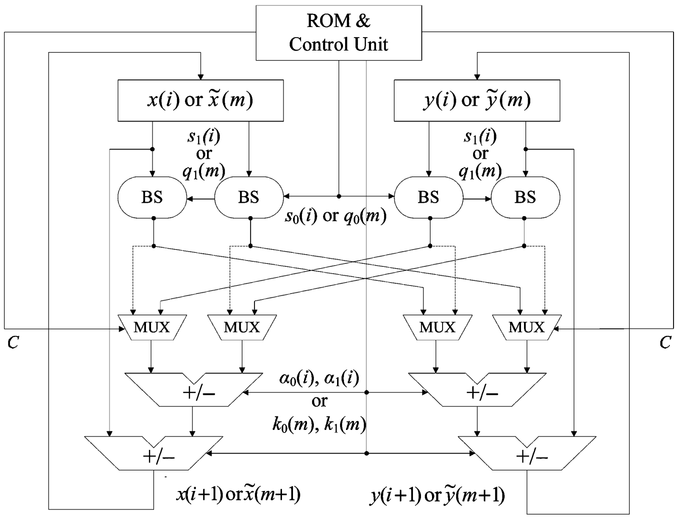

Fig. 3. EEAS-based CORDIC architecture. BS represents the Barrel Shifter, and C denotes the control signals for the micro-rotations.

EEAS scheme for the micro-rotation phase, a method has also been suggested in [21], as given below

$$\begin{aligned} \tilde{x}_{i+1} &= \tilde{x}_i + \left[k_1(i) \cdot 2^{-s_1(i)} + k_2(i) \cdot 2^{-s_2(i)}\right] \tilde{y}_i \\ \tilde{y}_{i+1} &= \tilde{y}_i + \left[k_1(i) \cdot 2^{-s_1(i)} + k_2(i) \cdot 2^{-s_2(i)}\right] \tilde{x}_i \quad (18) \end{aligned}$$

where $\tilde{x}_0 = x_{R_m}$ and $\tilde{y}_0 = x_{R_m}$. $k_1, k_2 \in \{-1, 0, 1\}$ and $q_1, q_2 \in \{1, 2, \ldots, n-1\}\}$.

The iterations for micro-rotation phase as well as the scaling phase could be implemented in the same architecture to reduce the hardware cost, as shown in Fig. 3.

3) Parallel Angle Recoding: The AR methods [19], [21] could be used to reduce the number of iterations by more than 50%, when the angle of rotation is known in advance. However, for unknown rotation angles, their hardware implementation involves more cycle time than the conventional implementation, which results in a reduction in overall efficacy of the algorithm. To reduce the cycle time of CORDIC iterations in such cases, a parallel angle selection scheme is suggested in [22], which can be used in conjunction with the AR method, to gain the advantages of the reduction in iteration count, without further increase in the cycle time. The parallel AR scheme in [22] is based on dynamic angle selection, where the elementary angles α_i can be tested in parallel and the direction for the micro-rotations can be determined quickly to minimize the iteration period. During each iteration, the residual angle ω, is passed to a set of n adder-subtractor units that compute $\Delta_i = (\omega - \sigma_i \cdot \alpha_i)$ for each elementary angle $\alpha_i = \arctan 2^{-i}$ in parallel and the differences Δ_i for $0 \le i \le n$ are then fed to a binary-tree like structure to compare them against each other to find the smallest

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

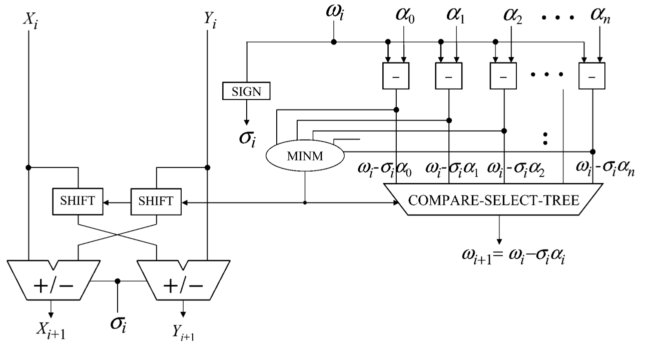
X_i
Y_i
ω_i
α_0 α_1 α_2 $\cdots$ α_n
SIGN
σ_i
MINM
$\omega_i-\sigma_i\alpha_0$ $\omega_i-\sigma_i\alpha_1$ $\omega_i-\sigma_i\alpha_2$ $\omega_i-\sigma_i\alpha_n$
SHIFT
SHIFT
COMPARE-SELECT-TREE
$\omega_{i+1}=\omega_i-\sigma_i\alpha_i$
+/−
+/−
X_{i+1}
σ_i
Y_{i+1}

Fig. 4. Architecture for parallel angle recoding.

difference. The $\sigma_i \cdot \alpha_i$ corresponding to the smallest difference $(\Delta_i)_{\min}$ is used as the angle of micro-rotation. The architecture for parallel angle recoding of [22] is shown in Fig. 4.

The parallel AR reduces the overall latency at the cost of high hardware-complexity of add/subtract-compare unit. For actual implementation, it is required to find a space-time trade-off and look at the relative performance in comparison with other approaches as well. The AR schemes based on EAS and EEAS however are useful for those cases where the angle of rotation is known in advance.

C. Hybrid or Coarse-Fine Rotation CORDIC

Based on the radix-2 decomposition, any rotation angle θ with n-bit precision could be expressed as a linear combination of angles from the set $\{2^{-i} : i \in \{1, 2, \ldots, n-1\}\}$, given by $\Sigma_{i=0}^{n-1} b_i 2^{-i}$, where $b_i \in \{0, 1\}$, explicitly specifies whether there is need of a micro-rotation or not. But, radix-2 decomposition is not used in the conventional CORDIC because that would not lead to simplicity of hardware realization. Instead, arctangents of the corresponding values of radix-2 based set are used as the elementary-angle-set with a view to implement the CORDIC operations only by shift-add operations. The key idea underlying the coarse-fine angular decomposition is that for the fine values of $\alpha_j = \arctan(2^{-j})$, (i.e., when $j > \lceil n/3 \rceil - 1$), $\tan(2^{-j})$ could be replaced by 2^{-j} in the radix-set for expansion of θ, since $\tan(2^{-j}) \approx 2^{-j}$ when j is sufficiently large.

1) Coarse–Fine Angular Decomposition: In the coarse-fine angular decomposition, the elementary-angle-set contains the arctangents of power-of-two for more-significant part while the less significant part contains the power-of-two values, such that the radix-set is given by $S = S_1 \bigcup S_2$, where $S_1 = \{\arctan 2^{-i} : i \in \{1, 2, \ldots, p-1\}\}$ and $S_2 = \{2^{-j} : j \in \{p, p+1, \ldots, n-1\}\}$, and j is assumed to be sufficiently large such that $\tan(2^{-j}) \to 2^{-j}$ [10]. For the hybrid decomposition scheme, the rotation angle could be partitioned into two terms expressed as

$$\theta = \theta_M + \theta_L \tag{19}$$

where θ_M and θ_L are said to be the coarse and fine subangles, respectively, given by

$$\theta_M = \sum_{i=1}^{p-1} \sigma_i \arctan 2^{-i} \quad \text{for } \sigma_i \in \{1, -1\} \tag{20a}$$

$$\theta_L = \sum_{i=p}^{n-1} d_i 2^{-i}, \quad \text{for } d_i \in \{0, 1\}. \tag{20b}$$

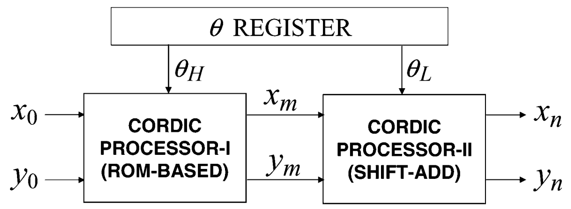

Fig. 5. Architecture for a Hybrid CORDIC algorithm [10].

A combination of coarse and fine micro-rotations are used in hybrid CORDIC operations in two cascaded stages. Coarse rotations are performed in stage-1 to have an intermediate vector

$$\begin{bmatrix} x_M \\ y_M \end{bmatrix} = \begin{bmatrix} 1 & -\tan(\theta_M) \\ \tan(\theta_M) & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{21}$$

and fine rotations are performed on the output of stage-1 to obtain the rotated output

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & -\tan(\theta_L) \\ \tan(\theta_L) & 1 \end{bmatrix} \begin{bmatrix} x_M \\ y_M \end{bmatrix}. \tag{22}$$

2) Implementation of Hybrid CORDIC: To derive the efficiency of hybrid CORDIC, the coarse and fine rotations are performed by separate circuits as shown in Fig. 5. The coarse rotation phase is performed by the CORDIC processor-I and the fine rotation phase is performed by CORDIC processor-II.

To have fast implementation, processor-I performs a pair of ROM look-up operations followed by addition to realize the rotation through angle θ_H. Since θ_L could be expressed as a linear combination of angels of small enough magnitude 2^{-j}, where $\tan(2^{-j}) \to 2^{-j}$, the computation of fine rotation phase can be realized by a sequence of shift-and-add operations. For implementation of the fine rotation phase, no computations are involved to decide the direction of micro-rotation, since the need of a micro-rotation is explicit in the radix-2 representation of θ_L. The radix-2 representation could also be recoded to express $\theta_L = \sum \tilde{b}_i 2^{-i}$ where $\tilde{b}_i \in \{-1, 1\}$ as shown in [9]. Since the direction of micro-rotations are explicit in such a representation of θ_L, it would be possible to implement the fine rotation phase in parallel for low-latency realization.

The hybrid decomposition could be used for reducing the latency by ROM-based realization of coarse operation. This can also be used for reducing the hardware complexity of fine rotation phase since there is no need to find the direction of micro-rotation. Several options are however possible for the implementation of these two stages. A form of hybrid CORDIC is suggested in [23] for very-high precision CORDIC rotation where the ROM size is reduced to nearly $n \cdot 2^{n/5}$ bits. The coarse rotations could be implemented as conventional CORDIC through shift-add operations of micro-rotations if the latency is tolerable.

3) Shift-Add Implementation of Coarse Rotation: Using the symmetry properties of the sine and cosine functions in different quadrants, the rotation through any arbitrary angle θ could be mapped from the full range $[0, 2\pi]$ to the first half the first quadrant $[0, \pi/4]$. The coarse-fine partition could be applied thereafter for reducing the number of micro-rotations necessary for fine rotations. To implement the course rotations

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

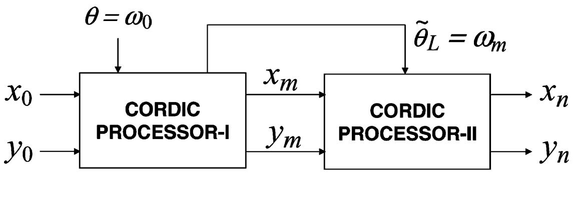

Fig. 6. Shift-add architecture for a Hybrid CORDIC algorithm.

through shift-add operations the coarse subangle θ_M is represented in [24] and [25] in terms of elementary rotations of the form $(\arctan 2^{-k})$ as

$$\theta_M = \sum_{i=2}^{n/3} 2^{-i} = \sum_{i=2}^{n/3} (\arctan 2^{-i} + \theta_{Li}) \tag{23}$$

where θ_{Li} is a correction term.

Using (23) on (19), one can find $\theta = \sum_{i=2}^{n/3} \arctan 2^{-i} + \tilde{\theta}_L$, where

$$\tilde{\theta}_L = \theta_L + \sum_{i=2}^{n/3} \theta_{Li}. \tag{24}$$

It is shown [25] that, based on the above decompositions using radix-2 representation, both coarse and fine rotations could be implemented by a sequence of shift-and-add operations in CORDIC iterations without ROM lookup table or the real multiplication operation. One such implementation is shown in Fig. 6. Processor-I performs CORDIC operations like that of conventional CORDIC for nearly the first one-third of the iterations and the residual angle as well as the intermediate rotated vector is passed to the processor-II. Processor-II can perform the fine rotation in one of the possible ways as in case of the circuit of Fig. 5.

The coarse-fine rotation approach in some modified forms has been applied for reduced-latency implementation of sine and cosine generation [24]–[28], high-speed and high-precision rotation [24], [26], and conversion of rectangular to polar coordinates and vice versa [29], [30].

4) Parallel CORDIC Based on Coarse-Fine Decomposition: In [31], the authors have proposed two angle recoding techniques for parallel detection of direction of micro-rotations, namely the binary to bipolar recoding (BBR) and micro-rotation angle recoding (MAR) to be used for the coarse part of the input angle θ_H. BBR is used to obtain the polarity of each bit in the radix-2 representation of θ_H to determine the rotation direction. MAR is used to decompose each positional binary weight $2^{-i}, \forall i, i = 1, 2, \ldots, m-1$ into a linear combination of arctangent terms. It is further shown in [32] that, the rotation direction can be decided once the input angle is known to enable parallel computation of the micro-rotations. Although the CORDIC rotation can be executed in parallel according to [32], the method for decomposition of each positional binary weight produces many extra stages of micro-rotation, especially when the bit-width of input angle increases. A more efficient recoding scheme has been proposed in [33] for the reduction of number of micro-rotations to be employed in parallel CORDIC rotations.

D. Redundant-Number-Based CORDIC Implementation

Addition/subtraction operations are faster in the redundant number system, since unlike the binary system, it does not involve carry propagation. The use of redundant number system is therefore another way to speed up the CORDIC iterations. A CORDIC implementation based on the redundant number system called as redundant CORDIC was proposed by Ercegovac and Lang and applied to matrix triangularization and singular value decomposition [34]. Rotation mode redundant CORDIC has been found to result in fast implementation of sinusoidal function generation, unitary matrix transformation, angle calculation and rotation [34]–[38]. Although redundant CORDIC can achieve a fast carry-free computation, the direction of the micro-rotation (the sign factor σ_i) cannot be determined directly unlike the case of the conventional CORDIC, since the redundant number system allows a choice $\sigma_i = 0$ along with the conventional choices 1 and -1 such that $\sigma_i \in \{-1, 0, 1\}$. Therefore, it requires a different formulation for selection of $\sigma_i = 0$, which is different for binary signed-digit representation and carry–save implementation. In radix-2 signed-digit representation, assuming—$(\Sigma_{k=i}^{\infty} \arctan 2^{-i}) \leq \omega_i \leq (\Sigma_{k=i}^{\infty} \arctan 2^{-i})$, it is shown that [6]

$$\sigma_i = \begin{cases} -1, & \text{if } \tilde{\omega}_i < 0 \\ 0, & \text{if } \tilde{\omega}_i = 0 \\ 1, & \text{if } \tilde{\omega}_i > 0 \end{cases} \tag{25}$$

where $\tilde{\omega}_i$ is the value of $2^j \omega_j$ truncated after the first fractional digit. Similarly for carry-save implementation, it is

$$\sigma_i = \begin{cases} -1, & \text{if } \tilde{\omega}_i < -1/2 \\ 0, & \text{if } \tilde{\omega}_i = 1/2 \\ 1, & \text{if } \tilde{\omega}_i > 1/2. \end{cases} \tag{26}$$

It can be noted from (25) and (26), that in some of the iterations no rotations are performed, so that the scale-factor becomes a variable which depends on the angle of rotation. Since the redundant CORDIC of [34] uses non-constant scale-factor, Takagi *et al.* [35] have proposed the double-rotation method and correcting-rotation method to keep the value of scale-factor constant. In double rotation method, in each iteration two micro-rotations are performed, such that when $\sigma_i = 0$, one positive and one negative micro-rotations are performed, and when $\sigma_i = +1$ or -1, respectively, two positive or two negative micro-rotations are performed. The scale-factor is retained constant in this case since the number of micro-rotations is fixed for any rotation angle but it doubles the iteration count. The correcting-rotation method examines the sign of $\tilde{\omega}_i$ constituted by some most significant digits of ω_i, and if $\tilde{\omega}_i \neq 0$ then σ_i is taken to be $\text{sign}(\tilde{\omega}_i)$ and σ_i is taken to be $+1$ otherwise. It is shown that the error occurring in this algorithm could be corrected by repetition of the iterations for $i = p, 2p, 3p \ldots$, etc., where p is the size of $\tilde{\omega}_i$. The branching CORDIC was proposed in [36] for fast on-line implementation for redundant CORDIC with a constant scale factor. The main drawback of this method, however, is its necessity of performing two conventional CORDIC iterations in parallel, which consumes more silicon area than classical methods [39]. The work proposed in [34] has also been extended to the vectoring mode [37], and correcting operations

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

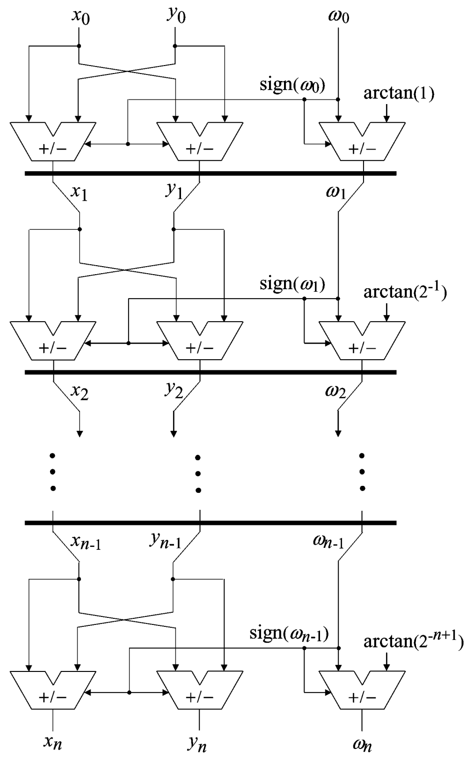

Fig. 7. Pipelined architecture for conventional CORDIC.

are included further to keep the scaling factor constant so as to eliminate the hardware for scaling.

E. Pipelined CORDIC Architecture

Since the CORDIC iterations are identical, it is very much convenient to map them into pipelined architectures. The main emphasis in efficient pipelined implementation lies with the minimization of the critical path. The earliest pipelined architecture that we find was suggested by Deprettere, Dewilde and Udo in 1984 [14]. Pipelined CORDIC circuits have been used thereafter for high-throughput implementation of sinusoidal wave generation, fixed and adaptive filters, discrete orthogonal transforms and other signal processing applications [40]–[44]. A generic architecture of pipelined CORDIC circuit is shown in Fig. 7. It consists of n stages of CORDIC units where each of the pipelined stages consists of a basic CORDIC engine of the kind shown in Fig. 2. Since the number of shifts to be performed by the shifters at different stages is fixed (shift-operation through i-bit positions is performed at the ith stage) in case of pipelined CORDIC the shift operations could be hardwired with adders; and therefore shifters are eliminated in the pipelined implementation. The critical-path of pipelined CORDIC thus amounts to the time required by the add/subtract operations in each of the stages. When three adders are used in each stage as shown in Fig. 7, the critical-path amounts to $T_{\mathrm{ADD}} + T_{\mathrm{MUX}} + T_{2C}$, where $T_{\mathrm{ADD}}, T_{\mathrm{MUX}}$ and T_{2C} are the time required for addition, 2:1 multiplexing and 2's complement operation, respectively. For known and constant angle rotations the sign of micro-rotations could be predetermined, and the need of multiplexing could be avoided for reducing the critical-path. The latency of computation thus depends primarily on the time required for an addition. Since there is very little room for reducing the critical path in the pipelined implementation of conventional CORDIC, digit-on-line pipelined CORDIC circuits based on the differential CORDIC (D-CORDIC) algorithm have been suggested to achieve higher throughput and lower pipeline latency.

TABLE III
DIFFERENTIAL CORDIC ALGORITHM

mode	algorithm
rotation mode	$x_{i+1} = x_i - \mathrm{sign}(\omega_i) \cdot 2^{-i} \cdot y_i$ $y_{i+1} = y_i + \mathrm{sign}(\omega_i) \cdot 2^{-i} \cdot x_i$ $\lvert\hat{\omega}_{i+1}\rvert = \lvert\lvert\hat{\omega}_i\rvert - \alpha_i\rvert$ $\mathrm{sign}(\omega_{i+1}) = \mathrm{sign}(\omega_i) \cdot \mathrm{sign}(\hat{\omega}_{i+1})$
vectoring mode	$\hat{x}_{i+1} = \hat{x}_i + \lvert\hat{y}_i\rvert \cdot 2^{-i}$ $\omega_{i+1} = \omega_i + \mathrm{sign}(y_i) \cdot \alpha_i$ $\mathrm{sign}(y_{i+1}) = \mathrm{sign}(y_i) \cdot \mathrm{sign}(\hat{y}_{i+1})$ $\lvert\hat{y}_{i+1}\rvert = \lvert\lvert\hat{y}_i\rvert - \hat{x}_i \cdot 2^{-i}\rvert$

F. Differential CORDIC Algorithm

D-CORDIC algorithm is equivalent to the usual CORDIC in terms of accuracy as well as convergence, but it provides faster and more efficient redundant number-based implementation of both rotation mode and vectoring mode CORDIC. It introduces some temporary variables corresponding to the CORDIC variables x, y and ω, that generically defined as

$$\hat{g}_{i+1} = \mathrm{sign}(g_i) \cdot g_{i+1} \tag{27}$$

which implies that $|\hat{g}_{i+1}| = |g_{i+1}|$ and $\mathrm{sign}(g_{i+1}) = \mathrm{sign}(g_i) \cdot \mathrm{sign}(\hat{g}_{i+1})$. The signs of g_i are, therefore, considered as being differentially encoded signs of $\hat{g}_i$ in the differential CORDIC algorithm [45]. The rotation and vectoring mode D-CORDIC algorithms are outlined in Table III.

D-CORDIC algorithm is suitable for efficient pipelined implementation which is utilized by Ercegovac and Lang [34] using on-line arithmetic based on redundant number system. Since the output data in the redundant on-line arithmetic can be available in the most-significant-digit-first (MSD-first) fashion, the successive iterations could be implemented by a set of cascaded stages, where processing time between the successive stages is overlapped with a single-digit time-skew, that results in a significant reduction in overall latency of computation. Moreover, in some redundant number representations, the absolute values and sign of the output are easily determined, e.g., in binary signed-digit (BSD) representation, the sign of a number corresponds to the sign of the first nonzero MSD, and negation of the number can be performed just by flipping signs of nonzero digits. A two-dimensional systolic D-CORDIC architecture is derived in [46] where phase accumulation is performed for direct digital frequency synthesis in the digit-level pipelining framework.

IV. SCALING, QUANTIZATION AND ACCURACY ISSUES

As discussed in Section II-A, scaling is a necessary operation associated with the implementation of CORDIC algorithm. Scaling in CORDIC could be of two types: 1) constant factor scaling and 2) variable factor scaling. In case of variable factor scaling the scale-factor changes with the rotation angle. It arises mainly because some of the iterations of conventional CORDIC are ignored (and that varies with the angle of rotation), as in

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

the case of higher-radix CORDIC and most of the optimized CORDIC algorithms. The techniques for scaling compensation for each such algorithms have been studied extensively for minimizing the scaling overhead. In case of conventional CORDIC, as given by (8), after sufficiently large number of iterations, the scale-factor K converges to ~ 1.6467605, which leads to constant factor scaling since the scale factor remains the same for all the angle of rotations. Constant factor scaling could be efficiently implemented in a dedicated scaling unit designed by canonical signed digit (CSD)-based technique [47] and common sub-expression elimination (CSE) approach [48], [49]. When the sum of the output of more than one independent CORDIC operations are to be evaluated, one can perform only one scaling of the output sum [50] in the case of constant factor scaling. In the following subsections, we briefly discuss some interesting developments on implementation of on-line scaling and realization of scaling-free CORDIC. Besides, we outline here the sources of error that may arise in a CORDIC design and their impact on implementation.

A. Implementation of Mixed-Scaling-Rotation

Dewilde *et al.* [51] have suggested the on-line scaling where shift-add operations for scaling and micro-rotations are interleaved in the same circuit. This approach has been used in [52] and improved further in [53]. In the mixed-scaling-rotation (MSR) approach, pioneered by Wu *et al.* [54]–[56], the micro-rotation and scaling phases are merged into a unified vector rotational model to minimize the overhead of the scaling operation [54]–[56]. The MSR-CORDIC can be applied to DSP applications, in which the rotation angles are usually known *a priori*, e.g., the twiddle factor in fast Fourier transform (FFT) and kernel components in other sinusoidal transforms. It is shown in [55] that the MSR technique can significantly reduce the total iteration count so as to improve the speed performance and enhance the signal-to-quantization-noise ratio (SQNR) performance by controlling the internal dynamic range. The MSR-CORDIC scheme has been applied to a variable-length FFT processor design [29], and found to result in significant hardware reduction in the implementation of twiddle-factor multiplications. Although, the interleaved scaling and MSR-CORDIC provide hardware reduction, they also lead to the reduction of throughput. For high-throughput implementation, one should implement the micro-rotations and scaling in two separate pipelined stages.

B. Low-Complexity Scaling

When the elementary angles pertaining to a rotation are "sufficiently small", defined by $\sin(\alpha_i) \cong \alpha_i = 2^{-i}$, and the rotations are only in one direction, the CORDIC rotation is given by the representation [57]

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \prod_{i=p}^{n-1} \begin{bmatrix} 1 - 2^{-(2i+1)} & 2^{-i} \\ -2^{-i} & 1 - 2^{-(2i+1)} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{28}$$

and $\omega_{i+1} = \omega_i - 2^{-i}$, (considering clockwise micro-rotations only), where x_i and y_i are the components of the vector after the ith micro-rotation, n is the input wordlength and $p = \lfloor (n - 2.585)/3 \rfloor$. The formulation of (28) performs the "actual" rotation where the norm of the vector is preserved at every micro-rotation.

However, the problem with this formulation is that the overall range of angles for which it can be used is very small, because, for 16-bit wordlength, the largest such angle is $\theta = \Sigma_{i=4}^{15}\alpha_i = \pm 7.16$, which obviously is quite small compared to the entire coordinate space. To overcome this problem, argument reduction is performed through "domain folding" [58] by mapping the target rotation-angles into the range $[0 \;\; \pi/8]$. Besides, the elementary rotations are carried out in an adaptive manner to enhance the rate of convergence so as to force the approximation error of final angle below a specified limit [59]. But, the domain-folding in some cases, involves a rotation through $\pi/4$ which demands a scaling by a factor of $1/\sqrt{2}$. Besides, the target range $[0 \;\; \pi/8]$ is still much larger than the range of convergence of the scaling-free realization. The formulation of (28), therefore, could be effectively used when a rotation through $\pi/4$ is not required and angles of rotations could be folded to the range $[-7.16^\circ \;\; 7.16^\circ]$. Generalized algorithms, and their corresponding architectures to perform the scale-factor compensation in parallel with the CORDIC iterations, for both rotation and vectoring modes are proposed in [60], where the compensation overhead is reduced to a couple of iterations. It is shown in [61] that since the scale-factor is known in advance, one can perform the minimal recoding of the bits of scaling-factor, and implement the multiplication thereafter by a Wallace tree. It is a good solution of low-latency scaling particularly for pipelined CORDIC architectures.

C. Quantization and Numerical Accuracy

Errors in CORDIC are mainly of two types: 1) the angle approximation error which originates from quantization of rotation angle represented by a linear combination of finite numbers of elementary angles and 2) the finite wordlength of the datapath resulting in the rounding/truncation of output that increases cumulatively through the successive iterations of micro-rotations. A third source of error that also comes into the picture results from the scaling of pseudo-rotated outputs. The scaling error is, however, also due to the use of finite wordlength in the scaling circuitry and is predominantly a rounding/truncation error. A detailed discussion on rounding error due to fixed and floating point implementations is available in [62]. In his earlier work, Walther [3] concluded that the errors in the CORDIC output are bounded, and $\log_2 n$ extra bits are required in the datapaths to take care of the errors. Hu [62] has provided more precise error bounds due to the angle approximation error for different CORDIC modes for fixed point as well as floating-point implementations. The error bound resulting for fixed point representation of arctangents is further analyzed by Kota and Cavallaro [63] and its impact on practical implementation has been discussed.

D. Area-Delay-Accuracy Trade-Off

Area, accuracy and latency of CORDIC algorithm depend mainly on the iteration count and its implementation. To achieve n-bit accuracy, if fixed-point arithmetic is applied, the wordlength of x and y data-path is $(n + 2 + \log 2(n))$ and for the computation of the angle θ, it is $(n + \log 2(n))$ [45],

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

TABLE IV
COMPUTATIONS USING CORDIC ALGORITHM IN DIFFERENT CONFIGURATIONS

operation	configuration	initialization	output	post-processing and remarks
$\cos\theta, \sin\theta, \tan\theta$	CC-RM	$x_0 = 1$ $y_0 = 0$ and $\omega_0 = \theta$	$x_n = \cos\theta$ $y_n = \sin\theta$	$\tan\theta = (\sin\theta / \cos\theta)$
$\cosh\theta, \sinh\theta$ $\tanh\theta, \exp(\theta)$	HC-RM	$x_0 = 1$ $y_0 = 0$ and $\omega_0 = \theta$	$x_n = \cosh\theta$ $y_n = \sinh\theta$	$\tanh\theta = (\cosh\theta / \sinh\theta)$ $\exp(\theta) = (\cosh\theta + \sinh\theta)$
$\ln(a), \sqrt{a}$	HC-VM	$x_0 = a + 1$ $y_0 = a - 1$ and $\omega_0 = 0$	$x_n = \sqrt{a}$ $\omega_n = \frac{1}{2}\ln(a)$	$\ln(a) = 2\omega_n$
$\arctan(a)$	CC-VM	$x_0 = a$ $y_0 = 1$ and $\omega_0 = 0$	$\omega_n = \arctan(a)$	no pre- or post-processing
division (b/a)	LC-VM	$x_0 = a$ $y_0 = b$ and $\omega_0 = 0$	$\omega_n = b/a$	no pre- or post-processing
polar-to-rectangular	CC-RM	$x_0 = R$ $y_0 = 0$ and $\omega_0 = \theta$	$x_n = R\cos\theta$ $y_n = R\sin\theta$	no pre- or post-processing
rectangular-to-polar $\tan^{-1}(b/a)$ and $\sqrt{a^2 + b^2}$	CC-VM	$x_0 = a$ $y_0 = b$ and $\omega_0 = 0$	$x_n = \sqrt{a^2 + b^2}$ $\omega_n = \arctan(b/a)$	no pre- or post-processing

The computation of $\tan\theta$ and $\tanh\theta$ require one division operation, while $\exp(\theta)$ and $\ln(a)$ require one addition and one shift, respectively, for post-processing. The computation of $\sqrt{a}$ and $\ln(a)$ require one increment and one decrement for pre-processing as shown in column 3, for "initialization".

[63]. The hardware requirement therefore increases accordingly with the desired accuracy. Floating-point implementation naturally gives higher accuracy than its fixed-point counterpart, but at the cost of more complex hardware. To minimize the angle approximation error, the smallest elementary angle α_{n-1} needs to be as small as possible [62]. This consequently demands more number of right-shifts and more hardware for the barrel-shifters and adders. Besides, to have better angle approximation, more number of iterations are required which increases the latency. The additional accuracy resulting from floating-point implementation or better angle approximation may not, however, be necessary in many applications. Thus, there is a need for trade-off between hardware-cost, latency and numerical accuracy subject to a particular application. Therefore, the designer has to check how much numerical accuracy is needed along with area and speed constraints for the particular application; and can accordingly decide on fixed or floating-point implementation and should set the wordlength and optimal number of iterations.

V. APPLICATIONS OF CORDIC

CORDIC technique is basically applied for rotation of a vector in circular, hyperbolic or linear coordinate systems, which in turn could also be used for generation of sinusoidal waveform, multiplication and division operations, and evaluation of angle of rotation, trigonometric functions, logarithms, exponentials and squareroot [6], [64], [65]. Table IV shows some elementary functions and operations that can be directly implemented by CORDIC. The table also indicates whether the coordinate system is circular (CC), linear (LC), or hyperbolic (HC), and whether the CORDIC operates in rotation mode (RM) or vectoring mode (VM), the initialization of the CORDIC and the necessary pre- or postprocessing step to perform the operation. The scale factors are, however, obviated in Table IV for simplicity of presentation. In this Section, we discuss how CORDIC is used for some basic matrix problems like QR decomposition and singular-value decomposition. Moreover, we make a brief presentation on the applications of CORDIC to signal and image processing, digital communication, robotics and 3-D graphics.

A. Matrix Computation

1) QR Decomposition: QR decomposition of a matrix can be performed through Givens rotation [66] that selectively introduces zeros into the matrix. Givens rotation is an orthogonal transformation of the form

$$\begin{bmatrix} c & c \\ -s & c \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \qquad (29)$$

where $\theta = \tan^{-1}(b/a), c = \cos\theta, s = \sin\theta$ and $r = \sqrt{a^2 + b^2}$. The QR decomposition requires two types of iterative operations to obtain an upper-triangular matrix using orthogonal transformations. Those are: (i) to calculate the Givens rotation angle, and (ii) to apply the calculated angle of rotation to the rest of the rows. Circular coordinate CORDIC is a good choice to implement both these Givens rotations, where the first operation is performed by a VM CORDIC and the second one is performed by an RM CORDIC. The CORDIC-based QR decomposition can be implemented in VLSI with suitable area-time trade-off using a systolic triangular array, a linear array or a single CORDIC processor that is reconfigurable for rotation and vectoring modes of operations. A detail explanation of these architectures are available in [64], [67].

2) Singular Value Decomposition and Eigenvalue Estimation: Singular value decomposition of a matrix $\mathbf{M}$ is given by $\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\boldsymbol{\Sigma}$ is a diagonal matrix of singular values. For CORDIC-based implementation of SVD, it is decomposed into 2×2 SVD problems, and solved iteratively. To solve each 2×2 SVD problem, two-sided Givens rotation is applied to each of the 2×2 matrices to nullify the off-diagonal elements, as described in the following:

$$\begin{bmatrix} \cos\theta_l & -\sin\theta_l \\ \sin\theta_l & \cos\theta_l \end{bmatrix} \mathbf{M} \begin{bmatrix} \cos\theta_r & -\sin\theta_r \\ \sin\theta_r & \cos\theta_r \end{bmatrix} = \begin{bmatrix} \psi_1 & 0 \\ 0 & \psi_2 \end{bmatrix} \qquad (30)$$

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

where $\mathbf{M}$ is a 2×2 input matrix to be decomposed; and θ_l and θ_r are, respectively, the left and right rotation angles, calculated from the elements of $\mathbf{M}$ using the following two relations:

$$\begin{array}{l}\theta_l + \theta_r = \arctan([c+b]/[d-a]) \\ \theta_l - \theta_r = \arctan([c-b]/[d+a])\end{array} \quad \text{for } \mathbf{M} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{31}$$

CORDIC-based architectures for SVD using this method were developed by Cavallaro and Luk [68]. A simplified design of array processor for the particular case ($c = b$ i.e., $\theta_l = \theta_r$) was developed further by Delosme [69] for the symmetric Eigenvalue problem. In a relatively recent paper [70], Liu *et al.* have proposed an application-specific instruction set processor (ASIP) for the real-time implementation of QR decomposition and SVD where circular coordinate CORDIC is used for efficient implementation of both these functions.

B. Signal Processing and Image Processing Applications

CORDIC techniques have a wide range of DSP applications including fixed/adaptive filtering [8], and the computation of discrete sinusoidal transforms such as the DFT [50], [52], [71], [72], discrete Hartley transform (DHT) [53], [73], [74], discrete cosine transform (DCT) [75]–[78], discrete sine transform (DST) [76]–[78] and chirp Z-transform (CZT) [79]. The DFT, DHT, and DCT [80] of an N-point input sequence $\{x(l)$ for $l = 0, 1, \ldots, N-1\}$, in general, are given by

$$X(k) = \sum_{l=0}^{N-1} C(k,l) \cdot x(l), \quad \text{for } k = 0, 1, \ldots, N-1 \tag{32}$$

where the transform kernel matrix is defined as

$$C(k,l) = \begin{cases} \cos(2\pi kl/N) - j\sin(2\pi kl/N), & \text{for DFT} \\ \cos(2\pi kl/N) + \sin(2\pi kl/N), & \text{for DHT} \\ \cos(\pi k(2l+1)/2N), & \text{for DCT.} \end{cases}$$

The input sequence for the DFT is, in general, complex and the computation of (32) can be partitioned into blocks of form: $[\mathrm{Re}[x(l)] \cdot \cos(2\pi kl/N) \pm \mathrm{Im}[x(l)] \cdot \sin(2\pi kl/N)]$, which is in the same form as the output of RM-CORDIC, for $\theta = 2\pi kl/N$. In case of DHT similarly the computation can also be transformed into a $N/2$ computations of the form $[x(l) \cdot \cos(2\pi kl/N) \pm [x(N-l)] \cdot \sin(2\pi kl/N)]$ to be implemented efficiently by RM-CORDIC units. These features of DFT and DHT are used to design parallel and pipelined architectures for the computation of these two transforms [50], [52], [53], [71]–[74]. It is shown that [76], [77] by simple input-output modification, one can transform the DCT and DST kernels into the DHT form to compute then by rotation mode CORDIC. Similarly in [79], CZT is represented by a DFT-like kernel by simple pre-processing and post-processing operations, and implemented through CORDIC rotations. The CORDIC technique has also been used in many image processing operations like spatial domain image enhancement for contrast stretching, logarithmic transformation and power-law transformation, image rotation, and Hough transform for line detection [81], [82]. CORDIC implementation of some of these applications are discussed in [83], [84]. Several other signal processing applications are discussed in detail in [64], which we do not intend to repeat here.

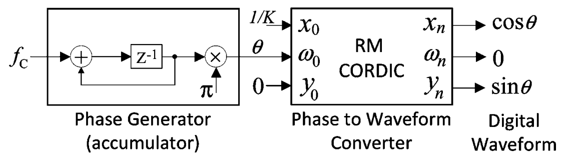

Fig. 8. CORDIC-based direct digital synthesizer. $\theta = \pi \cdot [\sum f_c]$.

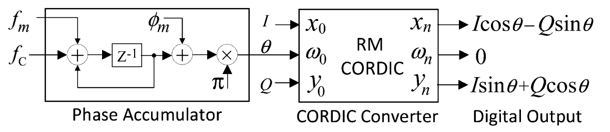

Fig. 9. A generic scheme to use RM CORDIC for digital modulation. I and Q are, respectively, the in-phase and quadrature signals to be modulated. $x_n = I \cdot \cos\theta - Q \cdot \sin\theta$, $y_n = I \cdot \sin\theta + Q \cdot \cos\theta$ and $\theta = [(\sum(f_c + f_m)) + \phi_m]\pi$.

C. Applications to Communication

CORDIC algorithm can be used for efficient implementation of various functional modules in a digital communication system [85]. Most applications of CORDIC in communications use the circular coordinate system in one or both CORDIC operating modes. The RM-CORDIC is mainly used to generate mixed signals, while the VM-CORDIC is mainly used to estimate phase and frequency parameters. We briefly outline here some of the important communication applications.

1) Direct Digital Synthesis: Direct digital synthesis is the process of generating sinusoidal waveforms directly in the digital domain. A direct digital synthesizer (DDS) (as shown in Fig. 8) consists of a phase accumulator and a phase-to-waveform converter [86], [87]. The phase-generation circuit increments the phase according to $[\sum f_c] \cdot \pi$, where f_c is the normalized carrier frequency in every cycle and feeds the phase information to the phase-to-waveform converter. The phase-to-waveform converter could be realized by an RM-CORDIC [88], [89], as shown in Fig. 8. The cosine and sine waveforms are obtained, respectively, by the CORDIC outputs x_n and y_n.

2) Analog and Digital Modulation: A generic scheme to use CORDIC in RM for digital modulation is shown in Fig. 9, where the phase-generation unit of Fig. 8 is changed to generate the phase according to $[(\sum(f_c + f_m)) + \phi_m] \cdot \pi$, for f_c and f_m being the normalized carrier and the modulating frequencies, respectively, and ϕ_M is the phase of modulating component. By suitable selection of the parameters f_c, f_m and ϕ_m and the CORDIC inputs x_0 and y_0, the generic scheme of Fig. 9 it could be used for digital realization of analog amplitude modulation (AM), phase modulation (PM), and frequency modulation (FM), as well as the digital modulations, e.g., amplitude shift keying (ASK), phase-shift keying (PSK), and frequency-shift keying (FSK) modulators. It could also be used for the up/down converters for quadrature-amplitude modulators (QAM) and full mixers for complex signals or phase and frequency corrector circuits for synchronization [85].

3) Other Communication Applications: By operating the CORDIC in vectoring mode, one can compute the magnitude and the angle of an input vector. The magnitude computation can be used for envelope-detection in an AM receiver or to detect FSK signal if it is placed after mark or space filters [90]. The angle computation in VM CORDIC, on the other hand, can

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

be used to detect FM and FSK signals and to estimate phase and frequency parameters [91]. A single VM-CORDIC can be used to perform these computations for the implementation of a slicer for a high-order constellation like the 32-APSK used in DVB-S2.

CORDIC circuits operating in both modes are also required in digital receivers for the synchronization stage to perform a phase or frequency estimation followed by a correction stage. This can be done by using two different CORDIC units, to meet the high speed requirement in Costas loop for phase recovery in a QAM modulation [92], [93]. On the other hand the burst-based communication system that needs a preamble for synchronization purposes, e.g., in case of IEEE 802.11a WLAN-OFDM receivers, can use a single CORDIC unit configurable for both operating modes since the estimation and correction are not performed simultaneously [94], [95]. Apart from these, the CORDIC-based QR decomposition has been used in multi-input-multi-output (MIMO) systems to implement V-BLAST detectors [96]–[98], and to implement a recursive-least-square (RLS) adaptive antenna beamformer [67], [99], [100].

D. Applications of CORDIC to Robotics and Graphics

Two of the key problems where CORDIC provides area and power-efficient solutions are: 1) direct kinematics and 2) inverse kinematics of serial robot manipulators. How CORDIC is applied in these applications is discussed below.

1) Direct Kinematics Solution (DKS) for Serial Robot Manipulators: A robot manipulator consists of a sequence of links, connected typically by either revolute or prismatic joints. For an N-degrees-of-freedom manipulator, there are N joint-link pairs with link 0 being the supporting base and the last link is attached with a tool. The joints and links are numbered outwardly from the base. The coordinates of the points on the ith link represented by (x_i, y_i, z_i) change successively for $i \in \{0, 1, \ldots, N-1\}$ due to successive rotations and translations of the links. The translation operations are realized by simple additions of coordinate values while the new coordinates of any point due to rotation are computed by RM-CORDIC circuits.

2) Inverse Kinematics for Robot Manipulators: The inverse kinematics problem involves determination of joint variables for a desired position and orientation for the tool. The CORDIC approach is valuable to find the inverse kinematic solution when a closed form solution is possible (when, in particular, the desired tool tip position is within the robot's work envelope and when joint angle limits are not violated). The authors in [101] present a maximum pipelined CORDIC-based architecture for efficient computation of the inverse kinematics solution. It is also shown [101], [102] that up to 25 CORDIC processors are required for the computation of the entire inverse kinematics solution for a six-link PUMA-type robotic arm. Apart from implementation of rotation operations, CORDIC is used in the evaluation of trigonometric functions and square root expressions involved in the inverse kinematics problems [103].

3) CORDIC for Other Robotics Applications: CORDIC has also been applied to robot control [104], [105], where CORDIC circuits serve as the functional units of a programmable CPU co-processor. Another application of CORDIC is for kinematics of redundant manipulators [106]. It is shown in [106] that the case of inverse kinematics can be implemented efficiently in parallel by computing pseudo-inverse through singular value decomposition. Collision detection is another area where CORDIC has been applied to robotics [107]. A CORDIC-based highly parallel solution for collision detection between a robot manipulator and multiple obstacles in the workspace is suggested in [107]. The collision detection problem is formulated as one that involves a number of coordinate transformations. CORDIC-based processing elements are used to efficiently perform the coordinate transformations by shift-add operations.

4) CORDIC for 3-D Graphics: The processing in graphics such as 3-D vector rotation, lighting and vector interpolation are computation-intensive and are geometric in nature. CORDIC architecture is therefore a natural candidate for cost-effective implementation of these geometric computations in graphics. A systematic formulation to represent 3D computer graphics operations in terms of CORDIC-type primitives is provided in [108]. An efficient stream processor based on CORDIC-type modules to implement the graphic operations is also suggested in [108]. 3-D vector interpolation is also an important function in graphics which is required for good-quality shading [109] for graphic rendering. It is shown that the variable-precision capability of CORDIC engine could be utilized to realize a power-aware implementation of the 3-D vector interpolator [110].

VI. Conclusion

The beauty of CORDIC is its potential for unified solution for a large set of computational tasks involving the evaluation of trigonometric and transcendental functions, calculation of multiplication, division, square-root and logarithm, solution of linear systems, QR-decomposition, and SVD, etc. Moreover, CORDIC is implemented by a simple hardware through repeated shift-add operations. These features of CORDIC has made it an attractive choice for a wide variety of applications. In the last fifty years, several algorithms and architectures have been developed to speed up the CORDIC by reducing its iteration counts and through its pipelined implementation. Moreover, its applications in several diverse areas including signal processing, image processing, communication, robotics and graphics apart from general scientific and technical computations have been explored. Latency of computation, however, continues to be the major drawback of the CORDIC algorithm, since we do not have efficient algorithms for its parallel implementation. But, CORDIC on the other hand is inherently suitable for pipelined designs, due to its iterative behavior, and small cycle time compared with the conventional arithmetic. For high-throughput applications, efficient pipelined-architectures with multiple-CORDIC units could be developed to take the advantage of pipelineability of CORDIC, because the digital hardware is getting cheaper along with the progressive device-scaling. Research on fast implementation of shift-accumulation operation, exploration of new number systems for CORDIC, optimization of CORDIC for constant rotation have scope for further reduction of its latency. Another way to use CORDIC efficiently, is to transform the computational algorithm into independent segments, and to implement the individual segments by different CORDIC processors. With enhancement of its throughput and reduction of latency, it is expected that CORDIC would be useful for many high-speed and real-time applications. The area-delay-accuracy trade-off

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

for different advanced algorithms may be investigated in detail and compared with in future work.

REFERENCES

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Computers*, vol. EC-8, pp. 330–334, Sept. 1959.

[2] J. E. Volder, "The birth of CORDIC," *J. VLSI Signal Process.*, vol. 25, pp. 101–105, 2000.

[3] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. 38th Spring Joint Computer Conf.*, Atlantic City, NJ, 1971, pp. 379–385.

[4] J. S. Walther, "The story of unified CORDIC," *J. VLSI Signal Process.*, vol. 25, no. 2, pp. 107–112, June 2000.

[5] D. S. Cochran, "Algorithms and accuracy in the HP-35," *Hewlett-Packard J.*, pp. 1–11, Jun. 1972.

[6] J.-M. Muller, *Elementary Functions: Algorithms and Implementation*. Boston, MA: Birkhauser Boston, 2006.

[7] S.-F. Hsiao and J.-M. Delosme, "Householder CORDIC algorithms," *IEEE Trans. Computers*, vol. 44, no. 8, pp. 990–1001, Aug. 1995.

[8] E. Antelo, J. Villalba, and E. L. Zapata, "A low-latency pipelined 2D and 3D CORDIC processors," *IEEE Trans. Computers*, vol. 57, no. 3, pp. 404–417, Mar. 2008.

[9] D. Timmermann, H. Hahn, and B. J. Hosticka, "Low latency time CORDIC algorithms," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 1010–1015, Aug. 1992.

[10] S. Wang, V. Piuri, and J. E. E. Swartzlander, "Hybrid CORDIC algorithms," *IEEE Trans. Computers*, vol. 46, no. 11, pp. 1202–1207, Nov. 1997.

[11] S. Wang and E. E. Swartzlander, Jr., "Merged CORDIC algorithm," in *IEEE Int. Symp. on Circuits Syst. (ISCAS'95)*, 1995, vol. 3, pp. 1988–1991.

[12] B. Gisuthan and T. Srikanthan, "Pipelining flat CORDIC based trigonometric function generators," *Microelectron. J.*, vol. 33, pp. 77–89, 2002.

[13] S. Suchitra, S. Sukthankar, T. Srikanthan, and C. T. Clarke, "Elimination of sign precomputation in flat CORDIC," in *IEEE Int. Symp. on Circuits Syst., ISCAS'05*, May 2005, vol. 4, pp. 3319–3322.

[14] E. Deprettere, P. Dewilde, and R. Udo, "Pipelined CORDIC architectures for fast VLSI filtering and array processing," in *IEEE Int. Conf. on Acoust., Speech, Signal Process., ICASSP'84*, Mar. 1984, vol. 9, pp. 250–253.

[15] H. Kunemund, S. Soldner, S. Wohlleben, and T. Noll, "CORDIC processor with carry save architecture," in *Proc. ESSCIRC 90*, Sept. 1990, pp. 193–196.

[16] E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapatai, "High performance rotation architectures based on the radix-4 CORDIC algorithm," *IEEE Trans. Computers*, vol. 46, no. 8, pp. 855–870, Aug. 1997.

[17] P. R. Rao and I. Chakrabarti, "High-performance compensation technique for the radix-4 CORDIC algorithm," *Proc. IEE Comput. and Digital Techn.*, vol. 149, no. 5, pp. 219–228, Sep. 2002.

[18] E. Antelo, T. Lang, and J. D. Bruguera, "Very-high radix circular CORDIC: Vectoring and unified rotation/vectoring," *IEEE Trans. Computers*, vol. 49, no. 7, pp. 727–739, July 2000.

[19] Y. H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," *IEEE Trans. Comput.*, vol. 42, no. 1, pp. 99–102, Jan. 1993.

[20] Y. H. Hu and H. H. M. Chern, "A novel implementation of CORDIC algorithm using backward angle recoding (BAR)," *IEEE Trans. Comput.*, vol. 45, no. 12, pp. 1370–1378, Dec. 1996.

[21] C.-S. Wu, A.-Y. Wu, and C.-H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes," *IEEE Trans. Circuits Syst. II: Anal. Digital Signal Process.*, vol. 50, no. 9, pp. 589–601, Sep. 2003.

[22] T. K. Rodrigues and E. E. Swartzlander, "Adaptive CORDIC: Using parallel angle recoding to accelerate CORDIC rotations," in *40th Asilomar Conf. on Signals, Syst. and Computers, ACSSC'06*, Oct.–Nov. 2006, pp. 323–327.

[23] M. Kuhlmann and K. K. Parhi, "P-CORDIC: A precomputation based rotation CORDIC algorithm," *EURASIP J. Appl. Signal Process.*, vol. 2002, no. 9, pp. 936–943, 2002.

[24] D. Fu and A. N. Willson, Jr., "A high-speed processor for digital sine/cosine generation and angle rotation," in *Conf. Rec. 32nd Asilomar Conf. on Signals, Syst. & Computers*, Nov. 1998, vol. 1, pp. 177–181.

[25] C.-Y. Chen and W.-C. Liu, "Architecture for CORDIC algorithm realization without ROM lookup tables," in *Proc. 2003 Int. Symp. on Circuits Syst., ISCAS'03*, May 2003, vol. 4, pp. 544–547.

[26] D. Fu and A. N. Willson, Jr., "A two-stage angle-rotation architecture and its error analysis for efficient digital mixer implementation," *IEEE Trans.Circuits Syst. I: Reg. Papers*, vol. 53, no. 3, pp. 604–614, Mar. 2006.

[27] S. Ravichandran and V. Asari, "Implementation of unidirectional CORDIC algorithm using precomputed rotation bits," in *45th Midwest Symp. on Circuits Syst., 2002. MWSCAS 2002*, Aug. 2002, vol. 3, pp. 453–456.

[28] C.-Y. Chen and C.-Y. Lin, "High-resolution architecture for CORDIC algorithm realization," in *Proc. Int. Conf. on Commun., Circuits Syst., ICCCS'06*, June 2006, vol. 1, pp. 579–582.

[29] D. D. Hwang, D. Fu, and A. N. Willson, Jr., "A 400-MHz processor for the conversion of rectangular to polar coordinates in 0.25-$/mu$-m CMOS," *IEEE J. Solid-State Circuits*, vol. 38, no. 10, pp. 1771–1775, Oct. 2003.

[30] S.-W. Lee, K.-S. Kwon, and I.-C. Park, "Pipelined cartesian-to-polar coordinate conversion based on SRT division," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 54, no. 8, pp. 680–684, Aug. 2007.

[31] S.-F. Hsiao, Y.-H. Hu, and T.-B. Juang, "A memory-efficient and high-speed sine/cosine generator based on parallel CORDIC rotations," *IEEE Signal Process. Lett.*, vol. 11, no. 2, 2004.

[32] T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, "Para-CORDIC: Parallel CORDIC rotation algorithm," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 51, no. 8, 2004.

[33] T.-B. Juang, "Area/delay efficient recoding methods for parallel CORDIC rotations," in *IEEE Asia Pacific Conf. on Circuits Syst., APCCAS'06*, Dec. 2006, pp. 1539–1542.

[34] M. D. Ercegovac and T. Lang, "Redundant and on-line CORDIC: Application to matrix triangularization and SVD," *IEEE Trans. Computers*, vol. 39, no. 6, pp. 725–740, June 1990.

[35] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Trans. Computers*, vol. 40, no. 9, pp. 989–995, Sept. 1991.

[36] J. Duprat and J.-M. Muller, "The CORDIC algorithm: New results for fast VLSI implementation," *IEEE Trans. Computers*, vol. 42, no. 2, pp. 168–178, Feb. 1993.

[37] J.-A. Lee and T. Lang, "Constant-factor redundant CORDIC for angle calculation and rotation," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 1016–1025, Aug. 1992.

[38] N. D. Hemkumar and J. R. Cavallaro, "Redundant and on-line CORDIC for unitary transformations," *IEEE Trans. Computers*, vol. 43, no. 8, pp. 941–954, Aug. 1994.

[39] J. Valls, M. Kuhlmann, and K. K. Parhi, "Evaluation of CORDIC algorithms for FPGA design," *J. VLSI Signal Process. Syst.*, vol. 32, no. 3, pp. 207–222, Nov. 2002.

[40] D. E. Metafas and C. E. Goutis, "A floating point pipeline CORDIC processor with extended operation set," in *IEEE Int. Symp. on Circuits Syst., ISCAS'91*, June 1991, vol. 5, pp. 3066–3069.

[41] Z. Feng and P. Kornerup, "High speed DCT/IDCT using a pipelined CORDIC algorithm," in *12th Symp. on Computer Arithmetic*, July 1995, pp. 180–187.

[42] M. Jun, K. K. Parhi, G. J. Hekstra, and E. F. Deprettere, "Efficient implementations of pipelined CORDIC based IIR digital filters using fast orthonormal μ-rotations," *IEEE Trans. Signal Process.*, vol. 48, no. 9, 2000.

[43] M. Chakraborty, A. S. Dhar, and M. H. Lee, "A trigonometric formulation of the LMS algorithm for realization on pipelined CORDIC," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 52, no. 9, 2005.

[44] E. I. Garcia, R. Cumplido, and M. Arias, "Pipelined CORDIC design on FPGA for a digital sine and cosine waves generator," in *Int. Conf. on Electr. Electron. Eng., ICEEE'06*, Sept. 2006, pp. 1–4.

[45] H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundant implementation without correcting iterations," *IEEE Trans. Computers*, vol. 45, no. 3, pp. 307–318, Mar. 1996.

[46] C. Y. Kang and E. E. Swartzlander, Jr., "Digit-pipelined direct digital frequency synthesis based on differential CORDIC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 5, pp. 1035–1044, May 2006.

[47] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[48] O. Gustafsson, A. G. Dempster, K. Johansson, M. D. Macleod, and L. Wanhammar, "Simplified design of constant coefficient multipliers," *J. Circuits, Syst., Signal Process.*, vol. 25, no. 2, pp. 225–251, Apr. 2006.

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

[49] G. Gilbert, D. Al-Khalili, and C. Rozon, "Optimized distributed processing of scaling factor in CORDIC," in *3rd Int. IEEE-NEWCAS Conf.*, June 2005, pp. 35–38.

[50] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Trans. Computers*, vol. 23, no. C-10, pp. 993–1001, Oct. 1974.

[51] P. Dewilde, E. F. Deprettere, and R. Nouta, "Parallel and pipelined VLSI implementation of signal processing algorithms," in *VLSI and Modern Signal Processing*, S. Y. Kung, H. J. Whitehouse, and T. Kailath, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[52] K. J. Jones, "High-throughput, reduced hardware systolic solution to prime factor discrete Fourier transform algorithm," *Proc.IEE Computers and Digital Techn.*, vol. 137, no. 3, pp. 191–196, May 1990.

[53] P. K. Meher, J. K. Satapathy, and G. Panda, "Efficient systolic solution for a new prime factor discrete Hartley transform algorithm," *Proc. IEE Circuits, Devices & Syst.*, vol. 140, no. 2, pp. 135–139, Apr. 1993.

[54] Z.-X. Lin and A.-Y. Wu, "Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for scaling-free high-performance rotational operations," in *IEEE Int. Conf. on Acoust., Speech, Signal Process., ICASSP'03*, Apr. 2003, vol. 2, pp. 653–656.

[55] C.-H. Lin and A.-Y. Wu, "Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for high-performance vector rotational DSP applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 11, pp. 2385–2396, Nov. 2005.

[56] C.-L. Yu, T.-H. Yu, and A.-Y. Wu, "On the fixed-point properties of mixed-scaling-rotation CORDIC algorithm," in *IEEE Workshop on Signal Process. Syst.*, Oct. 2007, pp. 430–435.

[57] A. S. Dhar and S. Banerjee, "An array architecture for fast computation of discrete hartley transform," *IEEE Trans. Circuits Syst.*, vol. 38, no. 9, pp. 1095–1098, Sep. 1991.

[58] K. Maharatna, A. Troya, S. Banerjee, and E. Grass, "New virtually scaling free adaptive CORDIC rotator," *Proc. IEE Computers and Digital Techn.*, vol. 151, no. 6, pp. 448–456, Nov. 2004.

[59] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified virtually scaling free adaptive CORDIC rotator algorithm and architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 11, pp. 1463–1474, Nov. 2005.

[60] J. Villalba, T. Lang, and E. Zapata, "Parallel compensation of scale factor for the CORDIC algorithm," *J. VLSI Signal Process.*, vol. 19, no. 3, pp. 227–241, Aug. 1998.

[61] D. Timmermann, H. Hahn, B. J. Hosticka, and B. Rix, "A new addition scheme and fast scaling factor compensation methods for CORDIC algorithms," *Integration, the VLSI J.*, vol. 11, no. 1, pp. 85–100, Mar. 1991.

[62] Y. H. Hu, "The quantization effects of the CORDIC algorithm," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 834–844, Apr. 1992.

[63] K. Kota and J. R. Cavallaro, "Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special-purpose processors," *IEEE Trans. Computers*, vol. 42, no. 7, pp. 769–779, July 1993.

[64] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Process. Mag.*, vol. 9, no. 3, pp. 16–35, July 1992.

[65] F. Angarita, A. Perez-Pascual, T. Sansaloni, and J. Vails, "Efficient FPGA implementation of cordic algorithm for circular and linear coordinates," in *Int. Conf. on Field Programmable Logic and Appl.*, Aug. 2005, pp. 535–538.

[66] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[67] G. Lightbody, R. Woods, and R. Walke, "Design of a parameterizable silicon intellectual property core for QR-based RLS filtering," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 4, pp. 659–678, Aug. 2003.

[68] J. R. Cavallaro and F. T. Luk, "CORDIC arithmetic for a SVD processor," *J. Parallel and Distributed Computing*, vol. 5, pp. 271–290, 1988.

[69] J. M. Delosme, "A processor for two-dimensional symmetric eigenvalue and singular value arrays," in *IEEE 21th Asilomar Conf. on Circuits, Syst., and Computers*, Nov. 1987, pp. 217–221.

[70] Z. Liu, K. Dickson, and J. V. McCanny, "Application-specific instruction set processor for SoC implementation of modern signal processing algorithms," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 4, pp. 755–765, Apr. 2005.

[71] K. J. Jones, "2D systolic solution to discrete Fourier transform," *Proc. IEE Computers and Digital Techn.*, vol. 136, no. 3, pp. 211–216, May 1989.

[72] T.-Y. Sung, "Memory-efficient and high-speed split-radix FFT/IFFT processor based on pipelined CORDIC rotations," *Proc. IEE Vision, Image Signal Process.*, vol. 153, no. 4, pp. 405–410, Aug. 2006.

[73] L.-W. Chang and S.-W. Lee, "Systolic arrays for the discrete Hartley transform," *IEEE Trans. Signal Process.*, vol. 39, no. 11, pp. 2411–2418, Nov. 1991.

[74] P. K. Meher and G. Panda, "Novel recursive algorithm and highly compact semisystolic architecture for high throughput computation of 2-D DHT," *Electron. Lett.*, vol. 29, no. 10, pp. 883–885, May 1993.

[75] W.-H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. 25, no. 9, pp. 1004–1009, Sep. 1977.

[76] B. Das and S. Banerjee, "Unified CORDIC-based chip to realise DFT/DHT/DCT/DST," *Proc. IEE Computers and Digital Techniques*, vol. 149, no. 4, pp. 121–127, July 2002.

[77] J.-H. Hsiao, L.-G. Ghen, T.-D. Chiueh, and C.-T. Chen, "High throughput CORDIC-based systolic array design for the discrete cosine transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 3, pp. 218–225, June 1995.

[78] D. C. Kar and V. V. B. Rao, "A CORDIC-based unified systolic architecture for sliding window applications of discrete transforms," *IEEE Trans. Signal Process.*, vol. 44, no. 2, pp. 441–444, Feb. 1996.

[79] Y. H. Hu and S. Naganathan, "A novel implementation of chirp Z-transform using a CORDIC processor," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 2, pp. 352–354, Feb. 1990.

[80] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[81] R. C. Gonzalez, *Digital Image Processing*, 3rd ed. Upper Saddle River, N.J.: Prentice Hall, 2008.

[82] N. Guil, J. Villalba, and E. L. Zapata, "A fast Hough transform for segment detection," *IEEE Trans. Image Process.*, vol. 4, no. 11, pp. 1541–1548, Nov. 1995.

[83] S. M. Bhandakar and H. Yu, "VLSI implementation of real-time image rotation," in *Int. Conf. on Image Process.*, Sept. 1996, vol. 2, pp. 1015–1018.

[84] S. Sathyanarayana, S. R. Kumar, and S. Thambipillai, "Unified CORDIC based processor for image processing," in *15th Int. Conf. on Digital Signal Process.*, July 2007, pp. 343–346.

[85] J. Valls, T. Sansaloni, A. Perez-Pascual, V. Torres, and V. Almenar, "The use of CORDIC in software defined radios: A tutorial," *IEEE Commun. Mag.*, vol. 44, no. 9, 2006.

[86] L. Cordesses, "Direct digital synthesis: A tool for periodic wave generation (part 1)," *IEEE Signal Process. Mag.*, vol. 21, no. 4, 2004.

[87] J. Vankka, *Digital Synthesizers and Transmitters for Software Radio*. Dordrecht, Netherlands: Springer, 2005.

[88] J. Vankka, "Methods of mapping from phase to sine amplitude in direct digital synthesis," in *50th IEEE International Frequency Control Symposium*, Jun. 1996, pp. 942–950.

[89] F. Cardells-Tormo and J. Valls-Coquillat, "Optimisation of direct digital frequency synthesisers based on CORDIC," *Electron. Lett.*, vol. 37, no. 21, 2001.

[90] M. E. Frerking, *Digital Signal Processing in Communication Systems*. New York: Van Nostrand Reinhold, 1994.

[91] H. Meyr, M. Moeneclaey, and S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. New York: Wiley, 1998.

[92] F. Cardells, J. Valls, V. Almenar, and V. Torres, "Efficient FPGA-based QPSK demodulation loops: Application to the DVB standard," *Lectures Notes on Computer Sci.*, vol. 2438, pp. 102–111, 2002.

[93] C. Dick, F. Harris, and M. Rice, "FPGA implementation of carrier synchronization for QAM receivers," *J. VLSI Signal Process.*, vol. 36, pp. 57–71, 2004.

[94] M. J. Canet, F. Vicedo, V. Almenar, and J. Valls, "FPGA implementation of an IF transceiver for OFDM-based WLAN," in *IEEE Workshop on Signal Process. Syst., SIPS'04*, 2004, pp. 227–232.

[95] A. Troya, K. Maharatna, M. Krstic, E. Grass, U. Jagdhold, and R. Kraemer, "Low-power VLSI implementation of the inner receiver for OFDM-based WLAN systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 2, pp. 672–686, Mar. 2008.

[96] Z. Guo and P. Nilsson, "A VLSI architecture of the square root algorithm for V-BLAST detection," *J. VLSI Signal Process. Syst.*, vol. 44, no. 3, pp. 219–230, Sept. 2006.

[97] Z. Khan, T. Arslan, J. S. Thompson, and A. T. Erdogan, "Analysis and implementation of multiple-input, multiple-output VBLAST receiver from area and power efficiency perspective," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 11, pp. 1281–1286, Nov. 2006.

[98] F. Sobhanmanesh and S. Nooshabadi, "Parametric minimum hardware QR-factoriser architecture for V-BLAST detection," *Proc. IEE Circuits, Devices and Syst.*, vol. 153, no. 5, pp. 433–441, Oct. 2006.

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.

[99] C. M. Rader, "VLSI systolic arrays for adaptive nulling [radar]," *IEEE Signal Process. Mag.*, vol. 13, no. 4, pp. 29–49, July 1996.
[100] R. Hamill, J. V. McCanny, and R. L. Walke, "Online CORDIC algorithm and VLSI architecture for implementing QR-array processors," *IEEE Trans. Signal Process.*, vol. 48, no. 2, pp. 592–598, Feb. 2000.
[101] C. Lee and P. Chang, "A maximum pipelined CORDIC architecture for inverse kinematic position computation," *IEEE Trans. Robot. Autom.*, vol. RA-3, no. 5, pp. 445–458, 1987.
[102] R. Harber, J. Li, X. Hu, and S. Bass, "The application of bit-serial CORDIC computational units to the design of inverse kinematics processors," in *Proc. IEEE Int. Conf. on Robot. and Autom.*, 1988, pp. 1152–1157.
[103] C. Krieger and B. Hosticka, "Inverse kinematics computations with modified CORDIC iterations," *Proc. IEE Computers and Digital Techn.*, vol. 143, pp. 87–92, Jan. 1996.
[104] Y. Wang and S. Butner, "A new architecture for robot control," in *Proc. IEEE Int. Conf. on Robot. Autom.*, 1987, pp. 664–670.
[105] Y. Wang and S. Butner, "RIPS: A platform for experimental real-time sensory-based robot control," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 853–860, 1989.
[106] I. Walker and J. Cavallaro, "Parallel VLSI architectures for real-time kinematics of redundant robots," in *Proc. IEEE Int. Conf. on Robot. Autom.*, 1993, pp. 870–877.
[107] M. Kameyama, T. Amada, and T. Higuchi, "Highly parallel collision detection processor for intelligent robots," *IEEE J. Solid-State Circuits*, vol. 27, pp. 300–306, 1992.
[108] T. Lang and E. Antelo, "High-throughput CORDIC-based geometry operations for 3D computer graphics," *IEEE Trans. Computers*, vol. 54, no. 3, pp. 347–361, Mar. 2005.
[109] B. Phong, "Illumination for computer generated pictures," *Commun. ACM*, pp. 311–317, June 1975.
[110] J. Euh, J. Chittamuru, and W. Burleson, "CORDIC vector interpolator for power-aware 3D computer graphics," in *IEEE Workshop on Signal Process. Syst., SIPS'02*, Oct. 2002, pp. 240–245.

Pramod Kumar Meher (SM'03) received the M.Sc. and M.Phil. degrees in physics, in 1978 and 1981, respectively, and the Ph.D. degree in science, in 1996, all from Sambalpur University, Sambalpur, India.

Currently, he is a Senior Scientist with the Institute for Infocomm Research, Singapore. Prior to this assignment he was a visiting faculty with the School of Computer Engineering, Nanyang Technological University, Singapore. From 1997 to 2002, he was a Professor of Computer Application with Utkal University, Bhubaneswar, India, and from 1993 to 1997, he was a Reader in electronics with Berhampur University, Berhampur, India. His research interest mainly includes the design and optimization of dedicated and reconfigurable architectures for computation-intensive algorithms, arithmetic circuits, and algorithm-architecture co-design for digital signal processing and communication.

Dr. Meher is a Fellow of IET (UK) and of IETE (India). He is currently serving as Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and *The Journal of Circuits, Systems, and Signal Processing*. He was the recipient of Samanta Chandrasekhar Award for excellence in research in engineering and technology for the year 1999.

Javier Valls (M'02) received the telecommunication engineering degree from the Universidad Politecnica de Cataluna, Spain, and the Ph.D. degree in telecommunication engineering from the Universidad Politecnica de Valencia, Spain, in 1993 and 1999, respectively.

He is with the Department of Electronics at Universidad Politecnica de Valencia, Valencia, Spain, since 1993, where he currently is an Associate Professor. His current research interests include the design of FPGA-based systems, computer arithmetic, VLSI signal processing, and digital communications.

Tso-Bing Juang (M'99) received the Ph.D. degree in computer science and engineering from National Sun Yat-sen University, Kaoshiung, Taiwan, in 2004. Since 2006, he has been with the Department of Computer Science and Information Engineering, National Pingtung Institute of Commerce, Taiwan, and is currently an Assistant Professor. His research interests include computer arithmetic and VLSI systems.

Dr. Juang received the Best Thesis Award from Taiwan Xerox's foundation in 1995. Currently he serves as the reviewers of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON COMPUTERS, and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.

K. Sridharan (S'84-M'96-SM'01) received the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1995.

He was an Assistant Professor at the Indian Institute of Technology (IIT) Guwahati, India, from 1996 to 2001. Since June 2001, he has been with IIT Madras, Chennai, India, where he is presently an Associate Professor. He was a visiting staff member at NTU, Singapore, in 2000–2001 and 2006–2008. He has supervised three Ph.D. candidates, and holds one joint patent. He has published more than 60 papers in various international journals and conferences.

Dr. Sridharan received the Computer Engineering Division Prize for a paper published in the Journal of I.E. (India) in 2002. He was also a joint recipient of the IEEE Vincent Bendix Award.

Koushik Maharatna (M'02) received the M.Sc. degree in electronic science from Calcutta University, in 1995 and the Ph.D. degree from Jadavpur University, Calcutta, India, in 2002.

From 2000 to 2003, he was a Research Scientist with IHP, Frankfurt (Oder), Germany, where, he was mainly involved in the design of a single-chip modem for the IEEE 802.11a standard. He is with the Electronics Systems and Devices Group at the School of Electronics and Computer Science of the University of Southampton, U.K., as a Senior Lecturer since 2006. His research interests include development of VLSI architectures for DSP and communication applications, computer arithmetic, low-power digital design, analog signal processing, and CNN.

Dr. Maharatna has served as session chair, reviewer and review committee management member in several IEEE conferences and journals. He is currently a member of the Engineering and Physical Research Council (EPSRC) college in the U.K. He is also a member of VLSI System Application (VSA) Technical Committee of IEEE Circuits and Systems Society.

Authorized licensed use limited to: UNIVERSITY OF SOUTHAMPTON. Downloaded on September 15, 2009 at 13:19 from IEEE Xplore. Restrictions apply.