Modified Virtually Scaling-Free Adaptive CORDIC Rotator Algorithm and Architecture

Koushik Maharatna, Swapna Banerjee, *Senior Member, IEEE*, Eckhard Grass, Milos Krstic, and Alfonso Troya, *Member, IEEE*

***Abstract*—In this paper, we proposed a novel Coordinate Rotation DIgital Computer (CORDIC) rotator algorithm that converges to the final target angle by adaptively executing *appropriate* iteration steps while keeping the scale factor *virtually constant* and *completely predictable*. The new feature of our scheme is that, depending on the input angle, the scale factor can assume only two values, *viz.*, 1 and $1/\sqrt{2}$, and it is *independent* of the number of executed iterations, nature of iterations, and word length. In this algorithm, compared to the conventional CORDIC, a reduction of 50% iteration is achieved on an average without compromising the accuracy. The adaptive selection of the appropriate iteration step is predicted from the binary representation of the target angle, and no further arithmetic computation in the angle approximation datapath is required. The convergence range of the proposed CORDIC rotator is spanned over the entire coordinate space. The new CORDIC rotator requires 22% less adders and 53% less registers compared to that of the conventional CORDIC. The synthesized cell area of the proposed CORDIC rotator core is 0.7 mm^2 and its power dissipation is 7 mW in IHP in-house 0.25-μm BiCMOS technology.**

***Index Terms*—Coordinate rotation digital computer (CORDIC), digital signal processing (DSP), scaling-free CORDIC, vector rotation, very large-scale integration (VLSI).**

I. INTRODUCTION

THE Coordinate Rotation DIgital Computer (CORDIC) algorithm [1], [2] has been used for many years for efficient implementation of vector rotation operations in hardware. It is executed merely by table look-up, shift, and addition operations. Thus, the corresponding hardware can be implemented in very economic fashion. Subsequently, it has been applied for many performance demanding applications in digital signal processing (DSP), image processing, and video technology like fast Fourier transform (FFT) [3], [4], discrete Hartley transform (DHT) [4], [5], discrete cosine transform (DCT) [4], [6], discrete sine transform (DST) [4], Hough transform (HT) [7]–[9], [12], graphics application [10], [11], and motion vector estimation [12].

In essence, a CORDIC can be operated in two different modes: the rotation and the vectoring mode. In the former mode of operation, given a vector with initial coordinate $(x0, y0)$ and a target rotation angle $(z0)$, the objective is to compute the final coordinate $(x1, y1)$ through a series of backward and forward rotation of the vector in an iterative manner. In the vectoring mode, the objective is to compute the magnitude and the phase angle of a vector given its initial and final coordinates. Table I shows the different modes of CORDIC operations in different coordinate systems where K_h and K_c are two constants known as scale factors for the hyperbolic and circular coordinate systems, respectively. However, despite its attractiveness, the conventional CORDIC algorithm has some drawbacks, such as slow speed, requirement of compensation of a bulk scale factor, and limited convergence range.

These drawbacks catalyze the research on the realization of high-performance special purpose CORDIC processors, and performance enhancement in algorithm as well as in the circuit level have been suggested. On the algorithmic level, the speed-up is achieved first by cleverly recoding the target angle (assumed known), and, second, by repeating some elementary rotation steps while bypassing some *not actually needed* elementary rotation steps [13], [25]. Although these techniques lead to a significant reduction of the number of iterations, the requirement of an extensive search procedure to find the right sequence of elementary rotation makes them computationally intensive in nature. A significant speed-up on the circuit level can be achieved using pipelined architectures and redundant arithmetic [16]–[21]. However, the determination of the direction of vector rotation is not trivial in such a number system and a false sign detection (hence a false rotation) is highly probable. To minimize the probability and the effect of false sign detection, several methods have been suggested [16], [18], [22]. Unfortunately, either they require extra iterations or hardware overhead.

In CORDIC processing, a bulk scale factor is generated that needs to be compensated. As long as all of the iterations allowed by a certain word length are carried out, the scale factor remains a constant and can be compensated with minimal hardware. However, in principle, the vector rotation for the majority of the angles lying in the coordinate space can be carried out using a smaller number of iterations by optimally choosing appropriate elementary rotation steps. This essentially requires bypassing or repeating some of the CORDIC iterations as was proposed in [13] and [25]. However, in such a case, the scale factor

Manuscript received February 4, 2003; revised September 12, 2004. This paper was recommended by Associate Editor R. Chandramouli.

K. Maharatna is with the Department of Electrical and Electronic Engineering, University of Bristol, Bristol, BS8 1UB, U.K. (e-mail: Koushik.Maharatna@bristol.ac.uk).

S. Banerjee is with the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, India (e-mail: swapna@ece.iitkgp.ernet.in).

E. Grass and M. Krstic are with the Communication Systems Department, IHP, 15236 Frankfurt (Oder), Germany (e-mail: grass@ihp-microelectronics.com; krstic@ihp-microelectronics.com).

A. Troya is with Corporate Research System Technologies, Infineon Technologies AG, Munich, Germany (e-mail: Alfonso.Troya@infineon.com).

Digital Object Identifier 10.1109/TCSVT.2005.856908

TABLE I
FUNCTIONALITY OF THE GENERALIZED CORDIC

Mode of operation	Hyperbolic	Linear	Circular
$y \to 0$ (Vectoring)	$X1 = K_h\sqrt{(x0^2 - y0^2)}$ $Z1 = z0 + \tanh^{-1}(y0/x0)$ $\lvert\tanh^{-1}(y0/x0)\rvert \le 1.1182$	$x1 = x0$ $z1 = z0 + (y0/x0)$ $\lvert y0/x0\rvert \le 1$	$x1 = K_c\sqrt{(x0^2 + y0^2)}$ $z1 = z0 + \tan^{-1}(y0/x0)$ $\lvert\tan^{-1}(y0/x0)\rvert \le 1.7433$ (99.9°)
$z \to 0$ (Rotation)	$x1 = K_h[x0\cosh(z0) + y0\sinh(z0)]$ $y1 = K_h[x0\sinh(z0) + y0\cosh(z0)]$ $\lvert z0\rvert \le 1.1182$	$x1 = x0$ $y1 = y0 + x0z0$ $\lvert z0\rvert \le 1$	$x1 = K_c[x0\cos(z0) - y0\sin(z0)]$ $y1 = K_c[x0\sin(z0) + y0\cos(z0)]$ $\lvert z0\rvert \le 1.7433$ (99.9°)

no longer remains constant or predictable. The situation is much worse when the CORDIC works in an embedded system where other circuitry of that system (a typical example is an OFDM synchronizer for IEEE 802.11(a) standard [15]) generates data and supplies it to the CORDIC. In such a case, the CORDIC has no *a priori* information about the actual angle of rotation and, hence, the scale factor is completely unpredictable. For its compensation, hardware circuitry is needed that is as complex as the original CORDIC itself. To the best of our knowledge, there is *no existing scheme* reported to date that can keep the scale factor constant and predictable while at the same time adaptively executing the minimum number of elementary rotation steps.

Another inherent problem of the CORDIC is that its range of convergence is small. Methods to increase the range of convergence use preprocessing of the input quantity or repetition of certain iteration steps [23]. However, the former approach results in an overhead in hardware while the latter increases the total number of CORDIC iterations. Furthermore, the repetition of iteration steps results in a nonconstant scale factor.

In this paper, we propose an algorithmic-level improved scheme that is suitable for the *rotation mode* of the CORDIC (i.e., $z0 \rightarrow 0$) in a circular coordinate system. For the convenience of nomenclature, from this point onwards, we will refer to the rotation mode of CORDIC in a circular coordinate system as *CORDIC rotator* operation.

To tackle the scale factor compensation problem, the authors have earlier proposed a *scaling-free* CORDIC algorithm having a small range of convergence [4], [24], and its error performance was shown to be identical to the conventional CORDIC [4]. However, in a more general scenario like [15], this algorithm cannot be applied efficiently. This *scaling-free* CORDIC algorithm is used as the baseline of the present work. The new contribution of the current work is: we develop a novel CORDIC rotator algorithm that *adaptively* executes only the *actually needed* elementary rotation steps to converge to the final target angle while keeping the scale factor *virtually constant* (it can take the values 1 or $1/\sqrt{2}$ only) and *completely predictable*. The final scale factor only depends on the actual value of the target angle and not on the number of executed iterations. On average, this adaptive selection results in about 50% reduction of computations without compromising the accuracy. The convergence range of the proposed CORDIC is spanned over the entire coordinate space. A novel feature of our scheme is that it is possible to *eliminate* the *entire arithmetic* and associated hardware for the angle approximation datapath. This results in a significant reduction of hardware and power consumption. The proposed algorithm and its very large-scale integration (VLSI) architecture are guaranteed to work even when the CORDIC operates in an embedded system. We call the proposed algorithm *virtually scaling-free adaptive CORDIC rotator algorithm* mainly because of the nature of the scale factor. The rest of the paper is structured as follows. In Section II, a brief review of the rotation mode of the conventional CORDIC algorithm in a circular coordinate system and the *scaling-free* CORDIC algorithm developed in [4], [24] are provided. Section III outlines the method of extending the angular convergence range of the *scaling-free* CORDIC rotator which leads to the new *virtually scaling-free* CORDIC rotator algorithm. In Section IV, we describe the architecture of the pipelined CORDIC rotator based on the proposed algorithm and the different performance parameters of it are discussed in Section V. In Section VI, the design of the processor in IHP in-house 0.25-μm BiCMOS technology is described. Conclusions are drawn in Section VII.

II. BRIEF REVIEW OF THE CONVENTIONAL AND THE *SCALING-FREE* CORDIC

A. *Conventional CORDIC*

The rotation of a vector $[x0\ y0]^T$ in the Cartesian coordinate system can be described as (considering clockwise rotation)

$$\begin{bmatrix} x1 \\ y1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x0 \\ y0 \end{bmatrix} \tag{1}$$

where $[x1\ y1]^T$ is the final vector and θ is the target angle of rotation. In the CORDIC algorithm, θ is expressed as the summation of a decreasing sequence of elementary angles α_i so that

$$\theta = \sum_{i=0}^{b-1} \sigma_i \alpha_i \tag{2}$$

$$\alpha_i = \tan^{-1}(2^{-i}) \tag{3}$$

where b is the wordlength of the machine in which the operation is to be implemented and $\sigma_i \in \{1, -1\}$ is known as the direction of vector rotation for the ith iteration. Substituting (2) into (1) and using (3), one may write

$$\begin{bmatrix} x1 \\ y1 \end{bmatrix} = \prod_{i=0}^{b-1} \cos\alpha_i \begin{bmatrix} 1 & \sigma_i 2^{-i} \\ -\sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{4}$$

and

$$\sigma_i = \mathrm{Sign}\left[\theta - \sum_{r=0}^{i-1} \alpha_r\right] \tag{5}$$

$$z_{i+1} = z_i + \sigma_i 2^{-i}. \tag{6}$$

Equations (4)–(6) are the basic working equations of the CORDIC rotator operation where $[x_i\ y_i]^T$ and z_i are the intermediate result vector and the residual angle, respectively, at the beginning of the ith iteration step. From the hardware implementation point of view, this vector rotation is nothing but a sequence of shift-and-add operations. However, the final result requires a scaling by a factor $\prod_{i=0}^{b-1} \cos\alpha_i$ (K_c in Table I). The scale factor remains a machine constant as long as the index i runs through all of the values from 0 to $b-1$, i.e., when all of the allowed iteration steps are executed. However, if i changes in a different manner, i.e., if some of the allowed iterations are bypassed or repeated in order to achieve a faster convergence rate or a larger convergence range, the scale factor will not remain constant and, for its compensation, one requires extra hardware and comparable postprocessing cycles.

B. Scaling-Free CORDIC Algorithm

The detailed description of the *scaling-free* CORDIC algorithm and its error performance is provided in [4], [24]. Thus, for the sake of conciseness, here we will outline only the essential features of that algorithm.

Unlike the conventional CORDIC algorithm, in the *scaling-free* CORDIC algorithm, the final target angle is achieved by rotating the vector in one direction only. This means that the final target angle is approximated as a *pure summation* of the elementary angles. These elementary rotational angles α_i are chosen in such a manner that the length of the vector be always preserved during each of the elementary rotation and is given by

$$\sin\alpha_i \cong \alpha_i = 2^i \tag{7}$$

(using first-order approximation of sine series) and subsequently

$$\cos\alpha_i = 1 - 2^{-(2i+1)}. \tag{8}$$

The use of first-order approximation of sine series for defining α_i in (7) imposes the following restriction on the allowed values of iteration index i:

$$\left\lceil \frac{(b-2.585)}{3} \right\rceil \le i \le b-1 \tag{9}$$

since i can only take integer values. However, for practical purposes, the lower bound of i can be relaxed slightly and can be given by the following equation:

$$\left\lfloor \frac{(b-2.585)}{3} \right\rfloor = p \le i \le b-1. \tag{10}$$

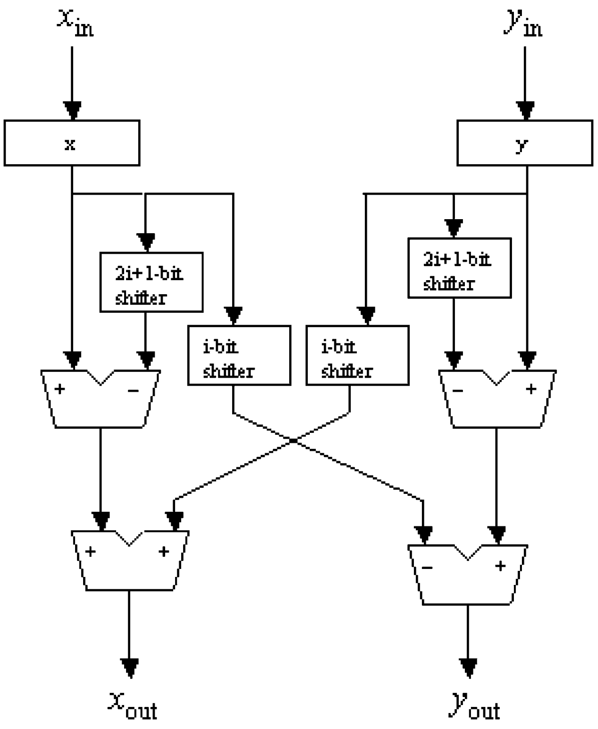


Fig. 1. Elementary rotational section for $i < b/2$.

Using (7) and (8), $\sigma_i = +1$ (since the sign of residual angle is always the same for all iteration), and clockwise rotation of the vector, (1) may be rewritten as

$$\begin{bmatrix} x1 \\ y1 \end{bmatrix} = \prod_{i=p}^{b-1} \begin{bmatrix} 1 - 2^{-(2i+1)} & 2^{-i} \\ -2^{-i} & 1 - 2^{-(2i+1)} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

$$z_{i+1} = z_i - 2^{-i}. \tag{11}$$

Equation (11) is the working equation of the *scaling-free* CORDIC rotator algorithm. It can be noted that, like (4), (11) can be realized in practice using only shift-and-add operations. On the other hand, contrary to (4), no scaling term appears in (11). The datapath component for implementing the operation stated in (11) is shown in Fig. 1. This can be viewed as an elementary rotational section rendering a rotation of α_i to its input vector $[x_{\text{in}}\ y_{\text{in}}]^T$. It requires four shifters, two subtractors, and two adder/subtractor in the datapath. Thus, compared to the datapath of an elementary rotational section of an *unscaled* CORDIC, it requires two shifters and two subtractors more. However, for the elementary rotational sections corresponding to $i \ge b/2$, these extra shifters and the subtractors can be omitted as the right shift of the input quantity by $(2i+1)$ becomes machine zero. Thus, the hardware cost for each of the elementary rotational sections corresponding to these values of i is exactly the same as that of the elementary rotational sections of the *unscaled* CORDIC. This simplified elementary rotational section is shown in Fig. 2.

Though the *scaling-free* CORDIC algorithm provides an elegant way to rule out the final scale factor, its biggest drawback is its extremely small convergence range and, thus, it cannot be considered as a general-purpose solution. Thus, methods have

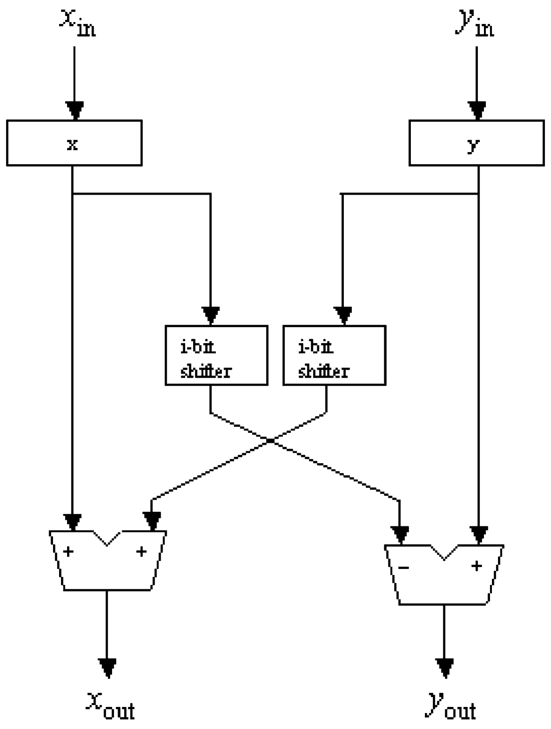


Fig. 2. Elementary rotational section for $i \geq b/2$.

to be developed to expand its angle computation range while preserving its scaling-free property.

III. New *Virtually Scaling-Free* CORDIC Rotator Algorithm

The main idea behind the development of a *virtually scaling-free* CORDIC rotator algorithm is to develop a CORDIC rotator algorithm having convergence over the entire coordinate space while staying within the general framework of the *scaling-free* CORDIC. The main requirement in such a case is to extend the convergence range of the *scaling-free* CORDIC itself. To do this, first an argument reduction technique is used to reduce the total angular range to be computed. Second, the elementary rotational operations are carried out in an adaptive manner to enhance the rate of convergence and to force the final angle approximation error below a certain prespecified limit. In the forthcoming discussion, this scheme is explained in detail.

The main objective of the argument reduction technique is to uniquely map the results of a CORDIC rotation with a large target angle θ to the results of a CORDIC rotation with a relatively small target angle ϕ. To do this, we divide the four quadrants of the coordinate system into 16 equal domains each having a uniform angular span of $\pi/8$ (i.e., four domains per quadrant). Any target angle (+ve or −ve) must lie in one of these 16 domains. We first examine the CORDIC rotation of an input vector with target angle θ lying in the first quadrant. This essentially means that θ lies in one of the four domains A ($[0,\pi/8)$), B ($[\pi/8,\pi/4)$), C ($[\pi/4,3\pi/8)$) and D ($[3\pi/8,\pi/2]$). In each domain, θ can be redefined in terms of another angle ϕ bounded in the interval $[0,\pi/8]$ by the following equations:

$$\theta = \phi, \text{ in domain A} \tag{12}$$

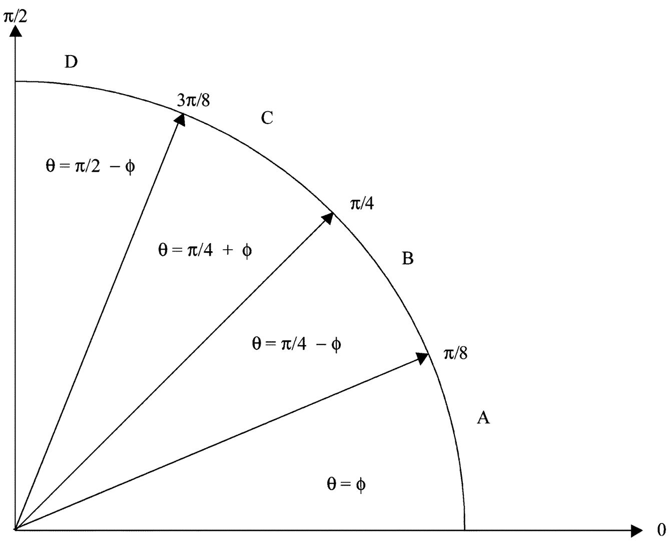


Fig. 3. Different domains and the definition of the modified target angle (ϕ) in each of them.

$$\theta = \frac{\pi}{4} - \phi, \text{ in domain B} \tag{13}$$
$$\theta = \frac{\pi}{4} + \phi \text{ in domain C} \tag{14}$$
$$\theta = \frac{\pi}{2} - \phi \text{ in domain D.} \tag{15}$$

This is shown in Fig. 3. Using (12)–(15) in (1), the CORDIC rotator operation on an input vector $[x\ y]^T$ in different domains can be expressed in terms of ϕ as follows:

$$\begin{bmatrix} x_{fA} \\ y_{fA} \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \text{ in domain A} \tag{16}$$
$$\begin{bmatrix} x_{fB} \\ y_{fB} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} (\cos\phi + \sin\phi) & (\cos\phi - \sin\phi) \\ -(\cos\phi - \sin\phi) & (\cos\phi + \sin\phi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \text{ in domain B} \tag{17}$$
$$\begin{bmatrix} x_{fC} \\ y_{fC} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} (\cos\phi - \sin\phi) & (\cos\phi + \sin\phi) \\ -(\cos\phi + \sin\phi) & (\cos\phi - \sin\phi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \text{ in domain C} \tag{18}$$
$$\begin{bmatrix} x_{fD} \\ y_{fD} \end{bmatrix} = \begin{bmatrix} \sin\phi & \cos\phi \\ -\cos\phi & \sin\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \text{ in domain D} \tag{19}$$

where x_{f*} denotes the final vector resulting from a CORDIC rotator operation with target angles lying in a certain domain indicated by $*$.

Now denoting $[x1_+\ y1_+]^T$ and $[x1_-\ y1_-]^T$ as the result of CORDIC rotation for angle ϕ and $-\phi$ respectively, (16)–(19) can be written as

$$x_{fA} = x1_-, \quad y_{fA} = y1_-, \quad \phi = \theta \tag{20}$$
$$x_{fB} = \frac{1}{\sqrt{2}}[x1_+ \quad + y1_+] \quad y_{fB} = \frac{1}{\sqrt{2}}[-x1_+ \quad + y1_+], \quad \phi = \frac{\pi}{4} - \theta \tag{21}$$
$$x_{fC} = \frac{1}{\sqrt{2}}[x1_- \quad + y1_-] \quad y_{fC} = \frac{1}{\sqrt{2}}[-x1_- \quad + y1_-], \quad \phi = \theta - \frac{\pi}{4} \tag{22}$$
$$x_{fD} = y1_+ \quad y_{fD} = -x1_+, \quad \phi = \frac{\pi}{2} - \theta. \tag{23}$$

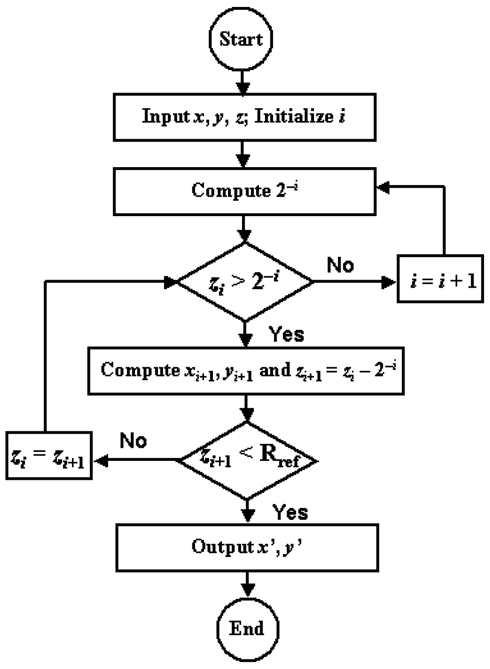
TABLE II
RESULTS OF THE CORDIC ROTATOR OPERATION FOR DIFFERENT QUADRANTS USING THE DOMAIN FOLDING TECHNIQUE

Range of the target angle	x1	y1
$[0, \pi/8)$	x_{fA}	y_{fA}
$[\pi/8, \pi/4)$	x_{fB}	y_{fB}
$[\pi/4, 3\pi/8)$	x_{fC}	y_{fC}
$[3\pi/8, \pi/2)$	x_{fD}	y_{fD}
$[\pi/2, 5\pi/8)$	y_{fA}	$-x_{fA}$
$[5\pi/8, 3\pi/4)$	y_{fB}	$-x_{fB}$
$[3\pi/4, 7\pi/8)$	y_{fC}	$-x_{fC}$
$[7\pi/8, \pi)$	y_{fD}	$-x_{fD}$
$[\pi, 9\pi/8)$	$-y_{fA}$	x_{fA}
$[9\pi/8, 5\pi/4)$	$-x_{fB}$	$-y_{fB}$
$[5\pi/4, 11\pi/8)$	$-x_{fC}$	$-y_{fC}$
$[11\pi/8, 3\pi/2)$	y_{fD}	$-x_{fD}$
$[3\pi/2, 13\pi/8)$	x_{fA}	y_{fA}
$[13\pi/8, 7\pi/4)$	$-y_{fB}$	x_{fB}
$[7\pi/4, 15\pi/8)$	$-y_{fC}$	x_{fC}
$[15\pi/8, 2\pi)$	x_{fD}	$-y_{fD}$

Thus, using the above equations, the CORDIC rotator operation with target angles lying in any domain in the first quadrant can be computed from the results of the CORDIC rotation with target angle ϕ (bounded in the interval $[0, \pi/8]$). This essentially means that the domains B, C, and D are effectively *folded back* to domain A. Hence, we call this technique *domain folding*. A consequence of the domain folding operation is the generation of a constant scale factor $1/\sqrt{2}$ for target angles lying in the domains B and C which can be realized with minimal hardware using only shift and add operations. The important point to be noted here is that this scale factor *does not depend* on the number of executed elementary rotations, nature of elementary rotation or the machine wordlength. On the other hand, for the target angle lying in domains A and D absolutely no scaling is required. Thus, the entire operation becomes *virtually scaling free*. It is apparent from (20) to (23) that for a particular target angle, rotation in only one direction (either $+$ve or $-$ve) is needed. Thus, the assumption that the target angle is approximated through only unidirectional sequence of vector rotation made for the *scaling-free* CORDIC is also valid here. This fact can be efficiently utilized to reduce the hardware complexity of the proposed CORDIC rotator during its architectural level implementation. We discuss this issue in Section IV.

By exploiting the symmetry of the coordinate axes, the domain folding technique can be extended to carry out CORDIC rotator operations with target angles lying in other quadrants as well. Table II provides a comprehensive report of the same, which shows that, depending on the quadrant in which the target angle lies, only the sign of the final vector components have changed. Thus, for computation of vector rotations with arbitrary target angles, the first step is to detect the quadrant and domain in which it lies. Once these parameters are detected, the computation can be carried out applying the appropriate equation from (20)–(23).

At this point, it is clear that, by using the *domain folding* technique, it is sufficient to consider the CORDIC operation for angles lying in the interval $[0, \pi/8]$ only. This results in a $16\times$ argument reduction. However, this range is still beyond the range of convergence of the original *scaling-free* CORDIC rotator unit. Thus, to take the full advantage of the *domain folding* technique, we need to expand the range of *scaling-free* CORDIC to $\pi/8$. This is done by choosing the iteration index i for each iteration *adaptively* in accordance with the residual angle still to be computed. Owing to the scaling-free property of the basic *scaling-free* CORDIC, this repetition of some elementary rotation steps does not generate any new scale factor. The flexibility offered in this approach is that the final error due to the angle approximation can be tailored by setting a prespecified limit of accuracy or number of iterations. The process of adaptive selection of i is described in the flowchart shown in Fig. 4 where, R_{ref} is the prespecified error limit (being adjusted by the user). In principle, the processor stops iterations when the residual angle becomes less than R_{ref}. If a large value for R_{ref} is chosen, the number of required iterations will be small but the accuracy will suffer. On the other hand, a very small value of R_{ref} results in a high accuracy but requires more iterations. Thus, the choice of R_{ref} should be done judiciously and according to the computational need of the target application.



Start
Input x, y, z; Initialize i
Compute 2^{-i}
$z_i > 2^{-i}$
No
$i = i + 1$
Yes
Compute x_{i+1}, y_{i+1} and $z_{i+1} = z_i - 2^{-i}$
$z_{i+1} < R_{\text{ref}}$
No
$z_i = z_{i+1}$
Yes
Output x', y'
End

Fig. 4. Process of adaptive selection of the iteration steps (searching algorithm).

Lemma: Only the iteration step corresponding to the smallest allowable value of i gets repeated more than once while the other values of i are nonrepetitive.

Proof: At the start of the jth iteration, let the residual angle be θ_r and $2^{-q} < \theta_r < 2^{-(q-1)} < 2^{-j}$, where $p < j < q$, where p is the smallest allowable value of i. Then, according to the flowchart shown in Fig. 4, the iteration steps (or the elementary rotation operations) corresponding to $i = j$ to $(q-1)$ will be bypassed and the elementary rotation corresponding to $i = q$ will be executed. After this elementary rotation operation, let the new residual angle θ_{rq} still be greater than 2^{-q}. This implies that another elementary rotation operation corresponding to $i = q$ is required (repetition). According to the proposed scheme, application of the rotation corresponding to $i = q$ twice (considering at least one repetition is required to bring down the value of residual angle less than 2^{-q}) essentially means that θ_r must be greater than or equal to $2^{-(q-1)}$. This is in contradiction to our initial assumption. Thus, it can be concluded that no elementary rotation steps corresponding to $i > p$ can get repeated.

However, if $(n+1)2^{-p} > \theta \geq n2^{-p}$, then the elementary rotation steps corresponding to $i = p$ will be executed n times.

The search algorithm employed for the adaptive selection of appropriate elementary rotational steps (shown in Fig. 4) is a simple comparison with 2^{-i} that can be implemented with minimal hardware. However, in the actual pipelined implementation this comparison is practically *not needed* owing to the one-sided rotation property of the proposed algorithm. In Section IV, we will describe this in more detail.

IV. Architecture of the CORDIC Rotator

The architecture of the new CORDIC rotator can be derived by a suitable hardware mapping of the algorithm described above. For sake of clarity, the implementation of a 16-b CORDIC rotator is described here as an example. All of the discussions presented in this section can be generalized for an n-bit CORDIC rotator as well.

For the design of the architecture, we have assumed that the decimal 1 is represented as 0 100 000 000 000 000. However, it is possible to choose a different definition also. Now, because of the *domain folding*, all of the angles lying in the coordinate space are effectively mapped into the range $[0, \pi/8]$. Thus, the effective maximum target angle that has to be computed is $\pi/8 = 0.392\,699$ rad. Using the definition of decimal 1 stated above, this angle can be represented in binary format as 0 001 100 100 100 010 with an error of $O(2^{-16})$. Thus, from the implementation point of view, for representing the absolute value of any angle lying in the modified convergence range one can omit the first three most significant bits (MSBs) and can use the 13 least significant bits (LSBs). In our architecture, we use this fact to reduce the total computation in the angle approximation datapath.

In principle, our design consists of three basic sections: the sign/domain detection circuitry, the basic CORDIC processor having convergence range of $[0, \pi/8]$, and the output circuitry. This is shown in Fig. 5.

Sign/Domain Detection Circuitry: The sign/domain detection circuitry (designated as sgn_detect in Fig. 5) has two 16-b-wide data word for $x0$ and $y0$ and an 18-b-data word for $z0$. We assume that the largest angle θ that can be assigned to the primary input $z0$ lies within the range $[0, 2\pi]$. According to our number representation, the value of 2π is 011001001000011111, which is an 18-b number. Keeping this fact in mind, we keep the 18-b word length for $z0$ input. Any negative angle will also fall within this range and thus can be translated to an appropriate positive angle. The principal job of the sign/domain detection circuitry is to detect the sign and domain of the target angle and subsequently, it applies *domain folding* technique to derive the 13-b unsigned representation of the modified target angle ϕ. It also generates two 2-b signals, namely *domain* and *quad*. While the *quad* signal indicates on which quadrant the original target angle lies, the *domain* signal indicates the corresponding domain in the first quadrant on

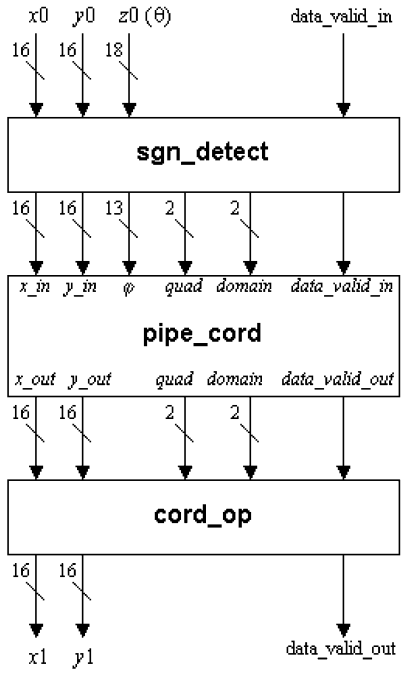


Fig. 5. Architecture of the complete CORDIC rotator.

which it has been folded back. The entire operation in this unit is performed in one clock cycle.

Basic Pipelined CORDIC Rotator Unit: The second section of our design is the basic CORDIC rotator unit having an internal wordlength of 16-b and a convergence range of $[0, \pi/8]$ (designated as pipe_cord in Fig. 5). The *scaling-free* CORDIC units shown in Figs. 1 and 2 acts as the elementary rotational sections in the present system. We implement the basic CORDIC rotator in pipelined fashion, thereby unfolding the iteration steps. With this approach, the shifters for each elementary rotational section become simple wire connections and effectively each elementary rotor section consumes a hardware area equivalent to four adder/subtractor units. On the other hand, as has been mentioned earlier, theoretically, for the elementary rotational sections corresponding to $i \geq 8$ (i.e., $b/2$), the hardware cost is equivalent to two adder/subtractor units. The allowed values of the CORDIC iteration in this case are $i = 4, 5, \ldots, 15$ [from (10)]. However, in our number representation scheme, shifting of any operand by 15 b to the right results in the retention of the sign bit only. Thus, for practical purposes one can omit the elementary rotational section corresponding to $i = 15$ from the design. On the other hand, following the same argument the two $(2i+1)$-bit shifters and the corresponding adders can also be removed from the elementary rotational section $i = 7$. Thus, we have used four adder/subtractors for each of the elementary rotational sections corresponding to $i = 4, \ldots, 6$ and two adder/subtractors for each of the elementary rotational sections corresponding to $i = 7, \ldots, 14$. In the forthcoming section, we will show that these assumptions do not introduce higher numerical errors compared to that of the conventional CORDIC.

In the pipelined implementation of the basic CORDIC rotator unit, the elementary rotational sections corresponding to $i = 4$ is used six times whereas, each of the sections corresponding

TABLE III
RELATIONSHIP BETWEEN THE POSITION OF LOGIC "1" IN THE REPRESENTATION OF ϕ AND THE APPROPRIATE ELEMENTARY ROTATIONAL SECTIONS

Position of logic'1'	12	11	10	9	8	7	6	5	4	3	2	1	0
Active stage (i)	4, 4, 4, 4	4, 4	4	5	6	7	8	9	10	11	12	13	14

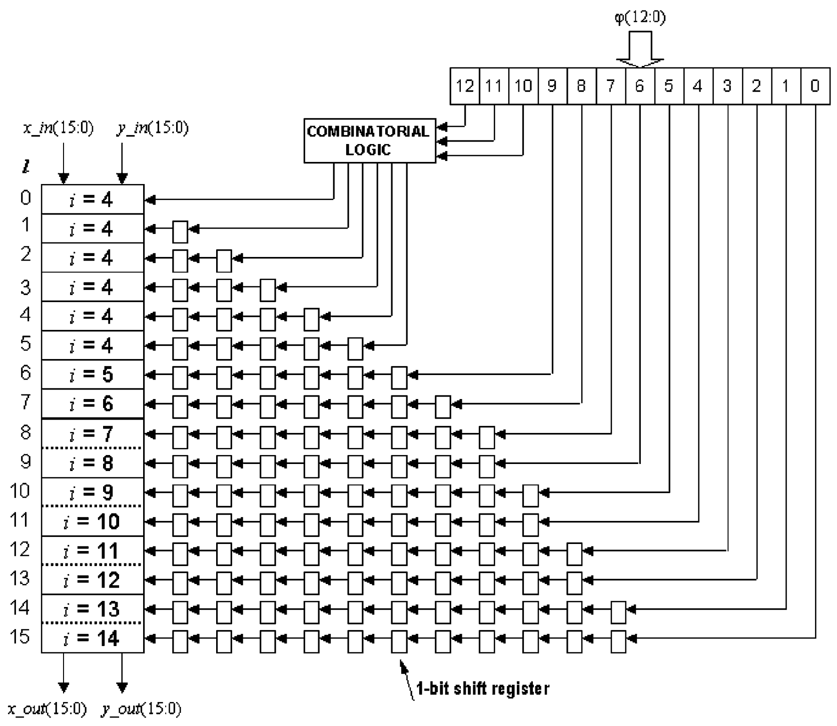


Fig. 6. Architecture of the basic CORDIC rotator pipeline.

to $i = 5, \ldots, 14$ is used once. The maximum angle that can be computed using such an arrangement is approximately 25° and thus covers our modified convergence range. Two's complement arithmetic is used for each of these elementary rotational sections. To make the pipeline completely balanced in terms of operating speed, we concatenate the elementary rotational sections corresponding to $i =$ (7, 8), (9, 10), (11, 12), and (13, 14), where, the elementary rotational sections within the parenthesis form a single pipeline stage each. With this special arrangement the basic CORDIC rotator pipeline becomes 12 stages long and the hardware complexity of each of these stages is equivalent to four 16-b adders.

The data and the information about the quadrant (signal *quad*) and domain (signal *domain*) of the initial vector detected in the sgn_detect module are transferred synchronously between two successive sections of the pipeline in a local register transfer manner. This means that each of the data in different sections of the pipeline has a *token* attributed to it that carries the information about the initial quadrant and domain of that particular data.

It has been mentioned earlier that in our algorithm, we approach the target angle by rotating the vector always in the same direction. Thus, in essence we approximate the target angle as a pure summation of 2^{-i}. As a result, the appropriate elementary rotational sections to be selected for a particular target angle have a one-to-one correspondence with the position of a logic "1" in the 13-b unsigned representation of ϕ. Table III summarizes the relationship between the position of a logic "1" in the representation of ϕ and the corresponding active elementary rotational sections. As an example, let us consider that $\phi = 20°$ (0.349 rad). The binary unsigned representation of this angle is 1 011 001 010 111. To achieve this target angle, the rotational sections those have to be activated are $i =$ 4, 4, 4, 4, 4, 5, 8, 10, 12, 13, and 14. The deactivated elementary rotational sections (corresponding to the bit position having logic "0") are bypassed. This implies a significant reduction of arithmetic computations that result in possible reduction of power consumption. It is to be noted that as the range of ϕ is $[0, \pi/8]$, under no condition a logic "1" can arise at 12th, 11th, and 10th bit position of the unsigned representation of ϕ at the same time. To keep the pipeline operation intact, we feed the individual bits of the 13-b unsigned representation of ϕ to the appropriate elementary rotational sections as an enable signal for that particular section through an array of single bit shift registers. The number of the shift registers corresponding to each section is chosen in such a manner that the appropriate section get enabled at the appropriate clock cycle. To enable the appropriate ones for the six $i = 4$ sections, we need a simple logic combination of the 12th, 11th, and 10th bits. For the other elementary rotational sections, the respective bits can be fed directly to the 0th shift register of the respective shift register array. The structure of the basic pipelined CORDIC processor using this arrangement is shown in Fig. 6 where l is the index of the elementary rotational sections of the pipeline. In Fig. 6, the solid lines indicate the boundary of each of the elementary rotational sections whereas the dotted lines indicate the concatenated elementary rotational stages.

This arrangement essentially mimics the search algorithm and eliminates the comparison of z_i with 2^{-i} and R_{ref} and the associated computation of new residual angle shown in Fig. 4. Thus, attendant hardware in the angle approximation datapath can be *omitted completely*. It is to be noted that, for the conventional CORDIC algorithm the target angle is approximated by a to-and-fro motion of the vector, and therefore the simple arrangement described here for the elimination of the arithmetic computation in the angle approximation datapath cannot be adopted there directly.

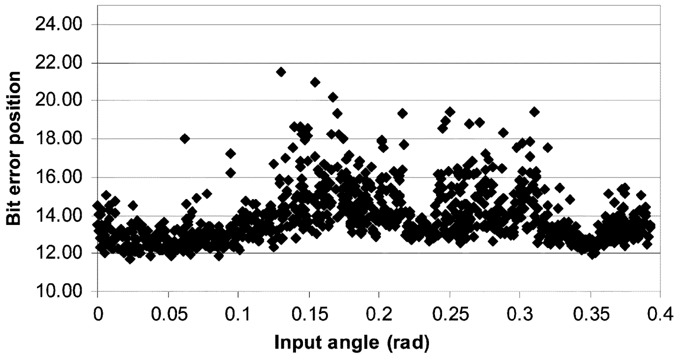
Output Unit: The last unit of the complete CORDIC rotator is the output unit designated as cord_op in Fig. 5. The circuit consists of two fixed scaling units of $1/\sqrt{2}$ and two adder/subtractor units. Each of the scaling units is realized using five 16-b adders. Thus, the overall hardware complexity of this unit is equivalent to twelve 16-b adders. Depending on the *domain* and *quad* signals this unit assigns the sign, applies either a scaling of $1/\sqrt{2}$ or simply passes the output vector emerging from the basic pipelined CORDIC it to the primary output [according to (20)–(23) and Table II]. All of the operations in the output unit are completed in one clock cycle.

V. Performance Evaluation and Comparison

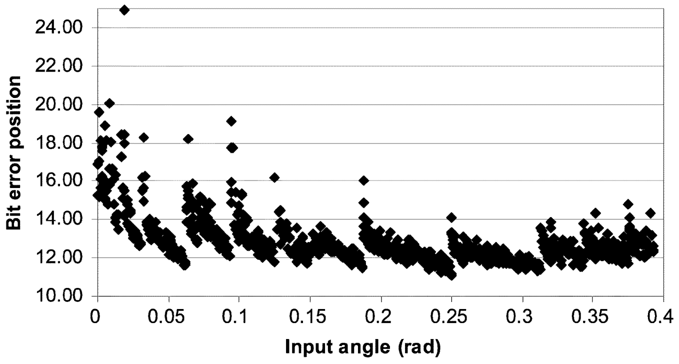
For evaluating the performance of the proposed CORDIC rotator algorithm and architecture, we mainly concentrate on two issues, error in the calculation of the x and y datapath and the area requirement for the complete CORDIC rotator. Here, we have not concentrated on the optimization of the number of adder/subtractors in the elementary rotational sections. Our main view is to consider the amount of hardware the proposed algorithm needs in its *natural* way. However, it is possible to reduce the number of adder/subtractors using different optimization schemes. Apart from these two issues, we also explore the possible advantage of the proposed architecture in terms of speed and power.

Error Analysis: In the implementation of a digital circuit, two main error sources exist: quantization of the input word and the truncation or cut-off rounding error due to the finite wordlength of arithmetic operations. However, for a CORDIC implementation, a third source of errors comes from how closely one can drive the $z0$ variable to zero, i.e., how closely the target angle is approximated. Since the proposed CORDIC rotator obeys the essential characteristics of the *scaling-free* CORDIC unit, its angle approximation error is also the same as the latter. Thus, in our case the angle approximation error is $O(2^{-16})$ [4].

However, the more dominant error in the hardware implementation of a CORDIC is the truncation error generated after each elementary rotation operation. This error gets accumulated at every section of the pipeline and dominates the final error in the calculation for the x and y values ($x1$ and $y1$, respectively, in our case).



(a)



(b)

Fig. 7. (a) Numerical errors in the x datapath. (b) Numerical errors in the y datapath.

To find out the combined effect of these three error mechanisms on the proposed architecture, we first generate a pseudorandom sequence of angles lying within our modified convergence range $[0, \pi/8]$. Using these angles as the inputs, the results for the x and y datapath of a Matlab model of ideal CORDIC function (1) is compared with the results for the x and y datapath of the VHDL coded proposed CORDIC rotator. The corresponding errors in terms of the decimal bit position are shown in Fig. 7. It is apparent from Fig. 7 that the upper bound of the errors for the x and y datapath is at the 12th decimal bit position. However, in order to achieve n-bit accuracy for the conventional CORDIC operation considering every form of quantization and truncation errors, one needs to use $(n + \log n + 2)$-bit data word length [14]. Thus, using 16-b word length, in case of the conventional CORDIC, the achievable upper bound of the error is approximately 10 b. This fact shows that the upper bound of the error resulting from the proposed CORDIC rotator method is *less* than that of the conventional CORDIC. However, the mean decimal bit position error in the proposed CORDIC rotator for the x and y datapath are at the 13th and 12th decimal bit position, respectively. This is due to the fact that, in case of the conventional CORDIC, the rotation operation is carried out at every elementary rotational sections, resulting in an accumulation of the error due to all the possible elementary rotational sections. On the other hand, in the proposed method, several of the elementary rotational sections are bypassed adaptively, and thus the final accumulated error is expected to be less than the former one on an average.

Area Requirement: In the proposed CORDIC rotator architecture, the basic CORDIC pipeline is 12 sections long and each of these sections requires four 16-b adders. The sign/domain detection unit needs two 16-b adders and two comparators. However, for both of the comparators, one of the input variables is constant. The hardware complexity of each of these comparators is estimated conservatively as equivalent to one 4-b adder. The output unit consists of two 16-b adders and two scaling units that are responsible for post-scaling by $1/\sqrt{2}$. Each of these scaling units consists of five 16-b adders. Thus, in our implementation the total area of the complete CORDIC rotator is approximately equivalent to 62 16-b adders $(4 \times 12 + 5 \times 2 + 2 + 2)$ and two 4-b adders. Considering one b-bit adder is equivalent to b full adders, the area of the proposed CORDIC rotator is equivalent to 1000 full adders.

In the conventional 16-b CORDIC implementation, the number of 16-b adders is 48 (including the angle approximation datapath). Here, we have assumed that the angle approximation datapath is 16 b wide like the x and y datapath. For compensation of the scale factor using shift-and-add technique one requires another 32 16-b adders. Thus, the complete conventional CORDIC structure needs 80 16-b adders altogether, which is equivalent to 1280 full adders. Thus, the proposed design requires approximately 22% less full adders compared to the conventional one. On the other hand, if the scale factor compensation network consists of two 16×16-b multiplier and each of them has 232 full adders ($n \times n$ multiplier requires $n \times (n - 2)$ full adders and n half adders) then the complete area of the conventional CORDIC processor is equivalent to 1232 full adders. This is still 19% higher compared to the proposed design.

In our implementation, the total number of registers required is 597. The number of registers that are required for the conventional CORDIC algorithm including scale factor compensation circuitry is 1280. Thus, our design needs 53% less registers compared to the conventional CORDIC.

Speed: The basic CORDIC rotator pipeline in our implementation is 12 stages long. Including the sign/domain detection section and the output section, the complete CORDIC pipeline length is 14 stages long. Thus, the latency of the processor is 14 clock cycles. The throughput of the proposed CORDIC rotator is one complete set of results/clock cycle. However, in this implementation, we do not need to carry out any arithmetic operation in the angle approximation datapath when the data is in the CORDIC pipeline. Thus, in this approach, one needs not to spend any time for the detection of sign of the direction of rotation (equivalently, sign of the residual angle) and subsequent approximation of the angle approximation datapath for a particular elementary rotational section. This implies that there is a possibility of significant speed enhancement of the processor.

Number of Iterations: In the proposed architecture, the appropriate elementary rotational sections are selected adaptively and the data processing is carried out only at those particular elementary rotational sections whereas, the other sections are bypassed. This operation in principle implies that, on an average, total number of iterations required for the proposed CORDIC rotator is smaller than that of the conventional CORDIC. To

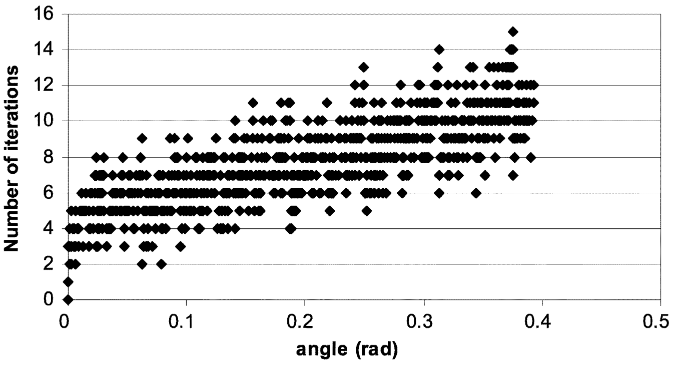


Fig. 8. Number of iterations required for the proposed CORDIC rotator for different target angles.

verify this, we once again used the pseudorandom angle sequence that lies within the modified convergence range generated earlier. The number of iterations corresponding to each of these angles is shown in Fig. 8. It is apparent from Fig. 8 that the maximum number of iterations (15) is needed for the angle 21.482° (0.3749 rad), where its unsigned binary representation is 1 011 111 111 111. For all other angles, the required number of iterations is smaller compared to that of the conventional CORDIC which is supposed to take 16 iterations (without scaling iterations) for each angle for the same order of accuracy. However, from Fig. 8, we calculated that the average number of iteration executed by the proposed CORDIC rotator is approximately 8, which is 50% less than that of the conventional CORDIC.

Power: The power consumption of a digital circuit is a function of silicon area and the number of arithmetic computations given a fixed frequency and operating voltage. It is already shown that the proposed CORDIC rotator requires less area compared to the conventional CORDIC and requires a smaller number of iterations to converge to the final target angle. The elimination of the angle approximation datapath results in a further reduction of arithmetic operations in the proposed CORDIC rotator. Thus, it can be expected that the architecture will consume less power compared to the conventional CORDIC processor.

Applicability of Redundant Arithmetic: For further speed up of the proposed CORDIC rotator, the redundant arithmetic can be used. Here, the x and y datapath can be represented in redundant format whereas, the target angle can be represented in conventional 2's complement format. Since the logic "1" in the unsigned binary representation of the modified target angle acts only as an enable signal for the appropriate rotational stages of the pipeline, the arithmetic computation (addition or subtraction) inside those stages can conveniently be carried out in constant time using redundant arithmetic. In this way, the principle problem of sign detection of the residual angle can be avoided while enjoying the high-speed advantage of it.

Comparison: The proposed CORDIC rotator algorithm is compared with two other similar type of algorithm described in [13] and [25]. The main results are shown in Table IV. The proposed one results in a variable number of iterations like the algorithm proposed in [13]. On the contrary, the algorithm described

TABLE IV
COMPARISON OF THE PROPOSED ALGORITHM WITH TWO OTHER SIMILAR ALGORITHMS

Name of the algorithms	Selection of rotational sequence	Elementary rotation	Search algorithm	Nature of required iterations	Nature of the scale factor
Convention CORDIC	{−1, 1}	Complete	No search	b (wordlength) Fixed	Fixed
[13]	{−1, 0, 1}	Selective	Complicated	N, Variable $N < b$	Variable
[25]	{−1, 0, 1}	Selective	Complicated	R, Fixed $R < b$	Variable
Proposed	{0, 1}	Selective	No explicit search	N, Variable $N < b$	1 or $1/\sqrt{2}$ (Fixed)

TABLE V
COMPARISON OF THE PROPOSED 16-b CORDIC ROTATOR ARCHITECTURE WITH SOME OTHER PUBLISHED REDUNDANT ARITHMETIC BASED ARCHITECTURES

CORDIC module	Equivalent number of full adders	Total number of registers
Conventional	768	768
Dawid [20]	1280	1984
Antelo [21]	896	1632
Proposed	768	533

in [25] uses a predefined fixed number of iterations. In all three cases, the number of executed iteration is less than that required for the conventional CORDIC algorithm. The principle advantage of the proposed algorithm is that, within the family of selective iteration, this is the *only one* that is *virtually scaling free*. The other algorithms require scale factor compensation owing to their selective nature. Thus, the final hardware requirement and the number of iterations including scaling iteration is minimum in the proposed algorithm. The second advantage of the proposed one is that it does not require complicated search algorithm like the others. In principle, the appropriate elementary rotational sections are automatically selected using the binary representation of the target angle. This results in significant hardware and power reduction compared to the others. Third, the proposed algorithm is guaranteed to work even when the target angle is not known in advance. Even for this case also, practically no search algorithm is needed. Regarding the error performance, the proposed algorithm gives same order of angle approximation error as that of the conventional CORDIC. Combining all of these facts, it is expected that the proposed algorithm will outperform the referenced ones in terms of speed, area, and power.

As a matter of interest, we also compare the hardware requirement of the proposed CORDIC rotator with some other redundant CORDIC implementation in Table V. For comparison on a uniform base, we have only considered the hardware requirement of the basic CORDIC pipeline without hardware for the scale factor compensation. It is apparent that the proposed one requires less number of equivalent full adders and registers compared to the others.

Drawbacks and General Discussions: Though the proposed algorithm eliminates the problem of adaptive selection of elementary rotation steps in conjunction with keeping the scale factor virtually constant, it is not also free of problem. The main problem is that the selection of largest elementary angle in this scheme depends on the wordlength [(10)]. This angle becomes increasingly smaller as the word length increases. Consequently, one needs to incorporate more sections of this elementary angle in the pipeline. As a result, the conventional CORDIC is expected to outperform the proposed one in terms of hardware requirement when the wordlength reaches 20 b. However, in that case, a hybrid scheme can be adopted to bring down the hardware cost. One may use some conventional CORDIC iteration (only unidirectional) to bring down the residual angle within the range of the *scaling-free* CORDIC iterations and then employ the proposed algorithm. In such a case, the scale-factor compensation circuitry required for those conventional CORDIC sections has to be integrated into the corresponding elementary rotational sections to avoid the generation of final scale factor and to maintain the *virtually scaling-free* property of the proposed algorithm. However, in general, hardware implementation with 16-b word length encompasses a vast application space, and for that the proposed CORDIC rotator shows significantly improved performance compared to the conventional CORDIC.

VI. IMPLEMENTATION OF THE CORDIC ROTATOR

For the design of the processor, the IHP in-house design kit is used. The CORDIC rotator is first modeled in VHDL and simulated using Mentor Graphics' Modelsim simulator. After the functional verification, Synopsys' Design Analyzer is used to synthesize the circuit for our in-house 0.25-μm BiCMOS technology with a target clock frequency of 20 MHz. The cell area of the complete processor after synthesis is 0.7 mm^2, which is equivalent to 24.7 k inverter gates in this technology. The power consumption reported by Synopsys' Design Analyzer is 7 mW at 2.5-V supply voltage.

After synthesis, the layout of the processor core is carried out using Cadence's Silicon Ensemble tool. The standard cell approach with a row utilization of 85% is deployed. The area of the processor core after layout is 0.9 mm^2. The Verilog netlist

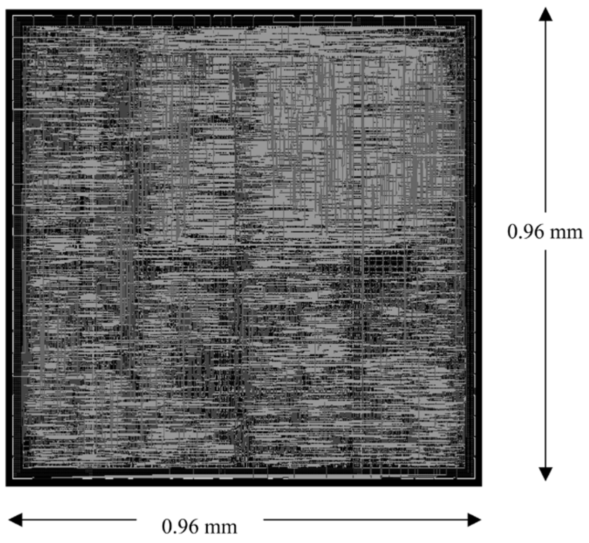


Fig. 9. Layout of the CORDIC rotor core.

extracted from the layout is resimulated including the Standard Delay File (SDF) using Modelsim, which confirms the correct behavior of the processor. The layout of the CORDIC rotator core is shown in Fig. 9.

VII. Conclusion

In this paper, we presented a novel algorithm and architecture of a special rotational CORDIC processor that operates in the circular coordinate system and has an unlimited angular convergence range. The algorithm *adaptively selects* the appropriate iteration steps and thus converges to the target angle executing a *minimum number* of iterations. On average, the number of iterations for the proposed method is about 50% less compared to the conventional CORDIC. The accuracy of the results is not compromised. The novel property of the proposed algorithm is that, unlike the conventional and previously reported CORDIC, the adaptive selection of the iteration steps has *no influence* on the final value of the scale factor, and thus it is possible to bypass the *actually not needed* iteration steps while keeping the scale factor *virtually constant*. Compared to the conventional CORDIC, in our scheme, the number of adders is reduced by 22%, and 53% fewer registers are needed. Moreover, the hardware requirement for the sign/domain detection circuitry is very small compared to that used for other argument reduction techniques.

Based on this algorithm, a 16-b pipelined CORDIC processor core was designed using IHP in-house 0.25-μm BiCMOS technology. The cell area of the CORDIC processor core is equivalent to 24.7 k inverter gates. The latency of the processor is 14 clock cycles. The power consumption of the CORDIC core reported by the synthesis tool is 7 mW. These figures show that the processor consumes little silicon area and is suitable for high-speed low-power applications. Currently, this CORDIC is used in an OFDM baseband processor for a wireless modem compliant with IEEE 802.11a [15].

References

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, Sep. 1959.

[2] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. Joint Spring Comput. Conf.*, vol. 38, Jul. 1971, pp. 379–385.

[3] E. F. Deprettere, P. Dewilde, and R. Udo, "Pipelined CORDIC architectures for fast VLSI filtering and array processing," in *Proc. ICASSP*, 1984, pp. 41 A 6.1–41 A. 6.5.

[4] K. Maharatna, A. S. Dhar, and S. Banerjee, "A VLSI array architecture for realization of DFT, DHT, DCT and DST," *Signal Process.*, vol. 81, pp. 1813–1822, 2001.

[5] E. L. Zapata and F. Arguello, "A VLSI constant geometry architecture for the Hartley and Fourier transform," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, no. 1, pp. 58–770, Jan. 1992.

[6] M. C. Mandal, A. S. Dhar, and S. Banerjee, "Multiplierless array architecture for computing discrete cosine transform," *Computers Elect. Eng.*, vol. 21, no. 4, pp. 327–333, 1994.

[7] J. D. Bruguera, N. Guil, T. Lang, J. Villalba, and E. L. Zapata, "CORDIC-based parallel/pipelined architecture for hough transform," *J. VLSI Signal Process.*, vol. 12, pp. 207–221, 1996.

[8] D. Timmermann, H. Hann, and B. J. Hosticka, "Hough transform using CORDIC method," *Electron. Lett.*, vol. 25, no. 3, pp. 205–206, 1989.

[9] K. Maharatna and S. Banerjee, "A VLSI array architecture for hough transform," *Pattern Recognit.*, vol. 34, pp. 1503–1512, 2001.

[10] H. Yoshimura, T. Nakanishi, and Y. Yamaguchi, "A 50 MHz CMOS geometrical mapping processor," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1360–1363, Oct. 1989.

[11] K. Maharatna and S. Banerjee, "CORDIC based array architecture for affine transformation of images," in *Proc. Int. Conf. Communications, Computers and Devices*, vol. II, Kharagpur, India, Dec. 2000, pp. 645–648.

[12] H. L. Li and C. Charkrabarti, "Hardware design of a 2-D motion estimation system based on hough transform," *IEEE Trans. Circuits Syst. II*, vol. 45, no. 1, pp. 80–95, Jan. 1998.

[13] 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.

[14] K. Kota and J. R. Cavallaro, "Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special purpose processors," *IEEE Trans. Comput.*, vol. 42, no. 4, pp. 769–779, Jul. 1993.

[15] M. Krstic, A. Troya, K. Maharatna, and E. Grass, "Optimized low-power synchronizer design for the IEEE 802.11(a) standard," in *Proc. ICASSP*.

[16] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Trans. Comput.*, vol. 40, no. 5, pp. 989–995, Sep. 1991.

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

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

[19] D. Timmermann and S. Dolling. Unfolded Redundant CORDIC VLSI Architectures With Reduced Area and Power Consumption. [Online]. Available: http://www-md.e-technik.uni-rostok.de/ma/dtim/vlsi97.pdf

[20] H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundant implementation without correcting iterations," *IEEE Trans. Comput.*, vol. 45, no. 2, pp. 307–318, Mar. 1996.

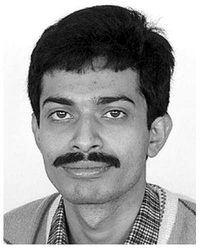
[21] E. Antelo, J. D. Bruguera, and E. L. Zapata, "Unified mixed radix 2–4 redundant CORDIC processor," *IEEE Trans. Comput.*, vol. 45, no. 5, pp. 1068–1073, Sep. 1996.

[22] D. S. Phatak, "Double step branching CORDIC : A new algorithm for fast sine and cosine generation," *IEEE Trans. Comput.*, vol. 47, no. 3, pp. 587–602, May 1998.

[23] X. Hu, R. G. Harber, and S. C. Bass, "Expanding the range of convergence of the CORDIC algorithm," *IEEE Trans. Comput.*, vol. 40, pp. 13–21, Jan. 1991.

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

[25] C. Wu and A. Wu, "Modified vector rotational CORDIC (MVR-CORDIC) algorithm and architecture," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 48, no. 6, pp. 548–561, Jun. 2001.



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

From 1996 to 2000, he was involved in projects sponsored by the Government of India undertaken at the Indian Institute of Technology (IIT), Kharagpur, India. From 2000 to 2003, he was a Research Scientist with IHP GmbH, Frankfurt (Oder), Germany. During this phase, his main involvement was related to the design of a single-chip modem for the IEEE 802.11a standard. In August 2003, he joined the Department of Electrical and Electronic Engineering, University of Bristol, Bristol, U.K., where he is currently a Lecturer. His research interests include development of VLSI architecture for application in digital signal processing and communication, computer arithmetic, low-power digital circuit design, analog signal processing, and CNN.

Dr. Maharatna has served as session chair for ISCAS 2005 and has acted as a reviewer for *Proceedings of the IEE Computer and Digital Technique*, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the *Journal of Signal Processing*, and *Signal Processing Letters*.

Swapna Banerjee (M'85–SM'99) received the B.E. and M.E. degree from Jadavpur University, Calcutta, India, and the Ph.D. degree from the Indian Institute of Technology (IIT), Kharagpur, India.

She performed her postdoctoral work at the University of Tokyo, Tokyo, Japan. In 1985, she joined as a Lecturer the Department of Electronics and Electrical and Computer Engineering, IIT, Kharagpur, where she is currently a Professor. She holds two patents and has authored/coauthored a number of publications in reputed international and national journals and conferences on various fields. She is also supervising a number of projects funded by national and international agencies. Her research interests span analog and digital VLSI design, device modeling, and embedded system design, especially biomedical instrumentation.

Eckhard Grass received the Dr.-Ing. degree in electronics from Humboldt University, Berlin, Germany, in 1992.

He was a Visiting Research Fellow with Loughborough University, Loughborough, U.K., from 1993 to 1995 and a Senior Lecturer in Microelectronics with the University of Westminster, London, U.K., from 1995 to 1999. Since 1999, he has been with IHP-GmbH, Frankfurt (Oder), Germany, leading a project on the implementation of a wireless broadband communication system. The aim of this project is to implement a complete 60-GHz, 1-Gb/s modem in SiGe:C BiCMOS technology. His research interests include data-driven (asynchronous) signal processing structures and low-power VLSI implementation of communication systems.

Milos Krstic was born in Nis, Serbia and Montenegro, in 1973. He received the Dipl.-Ing. and M.Sc. degrees in electronics from the University of Nis, Nis, Serbia and Montenegro, in 1997 and 2001, respectively. He is currently working toward the Dr.-Ing. degree at Brandenburg University of Technology, Cottbus, Germany.

In 2001, he joined IHP GmbH, Frankfurt (Oder), Germany, as a Research Associate in the Wireless Communication Systems Department. For the last few years, his work was mainly focused on low-power digital design for wireless applications and globally asynchronous locally synchronous (GALS) methodologies for large digital system integration.

Alfonso Troya was born in Barcelona, Spain, in 1975. He received the degree in telecommunications engineering from the Technical University of Catalonia, Barcelona, Spain, in 1999. and the Dr.-Ing. degree from Brandenburg University of Technology, Cottbus, Germany, in 2004.

The same year, he joined the Institute of Semiconductor Physics, IHP GmbH, Frankfurt (Oder), Germany, were he worked on the development and implementation of digital signal processing algorithms for broad-band wireless communication systems. At the end of 2004, he joined Infineon Technologies Corporate Research, Munich, Germany, where he is currently involved in the development of software-defined radio systems.

Dr. Troya is a member of the IEEE Signal Processing Society.