A Novel Implementation of CORDIC Algorithm Using Backward Angle Recoding (BAR)

Yu Hen Hu, *Senior Member, IEEE*, and Homer H.M. Chern

Abstract—We propose a *backward angle recoding* (*BAR*) method to eliminate redundant CORDIC elementary rotations and hence expedite the CORDIC rotation computation. We prove that for each of the linear, circular, and hyperbolic CORDIC rotations, the use of BAR guarantees more than 50% reduction of elementary CORDIC rotations provided the scaling factor needs not be kept constant. The proposed BAR algorithm is simple, and amenable to VLSI implementation. Taking practical applications into consideration, we discuss how to incorporate convergence range enhancement procedure with BAR, and how easy it is to devise a constant-scaling-factor BAR algorithm while still enjoying 25% reduction of CORDIC elementary rotations.

Index Terms—CORDIC, angle recoding, rotation based computing, VLSI, arithmetic unit, backward angle recoding.

1 Introduction

CORDIC [35], [6] (COordinate Rotation DIgital Computer), is an arithmetic algorithm for efficient evaluation of various transcendal and trigonometric functions. It has been incorporated in several arithmetic coprocessors [37]. Recently, it has also been applied to formulate and implement many modern digital signal processing (DSP) algorithms [1], [19], [21], [27], [32], [33], [34] which require the evaluation of complex number multiplications, elementary plane rotations, or lattice operations. In addition, efforts have been made to implement special purpose CORDIC processors for DSP applications [2], [3], [4], [5], [6], [7], [10], [11], [12], [15], [24], [30], [31], [32].

A distinct feature of the CORDIC algorithm is that it uses a sequence of *elementary rotations* to realize a variety of complicated, nonlinear elementary functions. Each elementary rotation requires two simple simultaneous shift-and-add operations to implement. By unfolding the iterations for elementary rotation, a pipelined CORDIC array processor can be realized [21] which will offer sustained high computing throughput rate. A reduction of the number elementary iterations for a specific algorithm, thus, will significantly reduce the latency (time between the availability of input data and available output), as well as hardware cost.

There are two different modes of operations in CORDIC: the *vector rotation* mode and the *angle accumulation* mode. In the vector rotation mode, the target rotation angle θ is given. The objective is to compute the final coordinate of a two by one vector given its initial coordinate. In the angle accumulation mode, the starting and ending coordinates are given. The objective is to compute the angle between the vectors pointing to these two coordinates.

- *Y.H. Hu is with the Department of Electrical and Computer Engineering, University of Wisconsin at Madison, Madison, WI 53706. E-mail: hu@engr.wisc.edu.*
- *H.H.M. Chern is with the Chung-Shang Institute of Science and Technology, Lung-Tan, Taoyuan, Taiwan, Republic of China.*

Manuscript received Aug. 16, 1995; revised Mar. 20, 1996.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C96166.

In the past, we have proposed a forward angle recoding (FAR) method [22] to reduce the number of elementary rotations required when the CORDIC algorithm is operated in the vector rotation mode. This FAR method has been applied to a number of DSP applications [19], [23]. In this paper, we report a completely different algorithm to reduce the number of required elementary rotations when the CORDIC algorithm is operated in the angle accumulation mode. We call this new algorithm *Backward Angle Recoding* (**BAR**) algorithm to distinguish it from the FAR algorithm.

The major contributions of this paper include

1) the derivation of the BAR algorithm;
2) a proof that using the BAR algorithm is guaranteed to save the number of elementary rotations by at least 50%;
3) discussions on implementation details of the BAR algorithm to show that it is an efficient implementation of the CORDIC algorithm.

The rest of the paper is organized as follows: In Section 2, the CORDIC algorithm is reviewed. In Section 3, the backward angle recoding algorithm is derived. The proof on the reduction of CORDIC iterations using the BAR algorithm is given in Section 4, together with discussions on convergence range extension, and constant scaling factor implementation. Simulation results are presented in that section, too.

2 Review of the CORDIC Algorithm

The CORDIC algorithm [35], [36] is a rotation based computing algorithm which performs generalized vector rotation in a linear ($m = 0$), circular ($m = 1$), or hyperbolic ($m = -1$) coordinate system. The rotation of an angle θ is accomplished via a sequence of *elementary* rotations. In each elementary rotation, a 2×1 vector $[x(i)y(i)]^t$ is rotated through an elementary rotation angle defined by

0018-9340/96$05.00 ©1996 IEEE

$$a_m(i) = \frac{1}{\sqrt{m}}\tan^{-1}\sqrt{m}2^{-s(m,i)} = \begin{cases}\tan^{-1}2^{-s(1,i)} & m = +1;\\ 2^{-s(0,i)} & m \to 0;\\ \tanh^{-1}2^{-s(-1,i)} & m = -1.\end{cases} \quad (1)$$

$\{s(m, i); i = 0, n-1\}$ is nondecreasing integer sequence. Due to the particular form of the elementary rotation angle, such an elementary rotation can be realized with simple shift and add operations.

The CORDIC iterations may be operated either in a forward rotation mode (also known as the vectoring mode [36]), or in a backward rotation mode (also known as the rotation mode [36]). In a forward rotation mode, the initial coordinate of a 2×1 vector $[x(0)y(0)]^t$ and the desired rotation angle θ are given. The objective is to find the final coordinate $[x'y']^t$ by rotating through the angle θ. In the backward rotation mode, the initial coordinate $[x(0)y(0)]^t$ is given. The goal is to compute the angle between $[x(0)y(0)]^t$ and the x axis where $y = 0$. The basic CORDIC algorithm can now be summarized as follows:

CORDIC Algorithm

Initiation: Given $[x(0)y(0)]^t$, $\theta(0)$[1], $\{\sigma(i); 0 \le i \le n-1\}$.

/* Elementary CORDIC Iterations */

For i = 0 to $n-1$ Do

$$\begin{bmatrix}x(i+1)\\ y(i+1)\end{bmatrix} = \begin{bmatrix}1 & m\mu(i)2^{-s(m,i)}\\ -\mu(i)2^{-s(m,i)} & 1\end{bmatrix}\begin{bmatrix}x(i)\\ y(i)\end{bmatrix} \quad (2)$$

$$\mu(i) = \begin{cases}sgn(\theta(i)) & \textit{forward rotation}\\ sgn(x(i))\cdot sgn(y(i)) & \textit{backward rotation}\end{cases}[1] \quad (3)$$

$$\theta(i+1) = \theta(i) - \mu(i)\cdot a_m(i);\quad 0 \le i \le n-1 \quad (4)$$

End

/* Scaling correction iterations */

For i = 0 to $n_s - 1$ Do

$$x(n+i+1) = x(n+i) + \sigma(i)2^{-r(m,i)}x(n) \quad (5a)$$

$$y(n+i+1) = y(n+i) + \sigma(i)2^{-r(m,i)}y(n) \quad (5b)$$

End /* scaling iterations */

Output: $x' = x(n+n_s)$, $y' = y(n+n_s)$, $\theta(n)$.

In this formulation, $sgn[x] = 1$ if $x > 0$; $= 0$ if $x = 0$; and $= -1$ if $x < 0$. It is not difficult to see that $\mu(i) = \pm 1$. Since

$$\sum_{i=0}^{n-1}|\mu(i)|$$

equals to the total number of elementary CORDIC rotations, in this paper, we will seek to reduce the number of CORDIC iterations by setting some $\mu(i) = 0$. The integer sequence $\{s(m, i); 0 \le i \le n-1\}$ is called a *shift sequence*, which impacts on the convergence and accuracy of the CORDIC algorithm. Usually, it is set $s(m, i) = i$ for $m = 0, 1$, and $s(-1, i) = i + 1$ for $m = -1$. We note that for $m = -1$, Walter has suggested to repeat several integers in the $\{s(m, i)\}$ sequence to ensure the convergence (accuracy) of the CORDIC iterations [36]. Different choices of the shift sequence have also been reported [1], [4], [8], [9].

The scaling operation is to compensate for the change of norm of $[x(i)y(i)]^t$ in the corresponding coordinate system during elementary CORDIC iterations. It requires multiplying $[x(n)y(n)]^t$ by a scaling factor S_m:

$$S_m = \prod_{i=0}^{n-1}S_m(i) \equiv \prod_{i=0}^{n-1}\frac{1}{\sqrt{1+\mu^2(i)\tan^2\sqrt{m}a_m(i)}} = \prod_{i=0}^{n-1}\frac{1}{\sqrt{1+m\mu^2(i)\cdot 2^{-2s(m,i)}}} \quad (6)$$

If $|\mu(i)| = 1$, S_m will be a constant and thus can be calculated ahead of time. In (5), S_m is represented in the usually format:

$$S_m = 1 + \sum_{i=1}^{n_s-1}\sigma(i)2^{-i}$$

Techniques such as *multiplier recoding* [13], [25] can be applied to reduce n_s, and hence the number of scaling iterations. Other methods to reduce overhead for scaling iterations have also been reported [1], [4], [8], [9].

The CORDIC algorithm has been used in many real time signal processing applications. It is often desirable, however, to expedite its execution. Past efforts have been focused on the reduction of number of scaling iterations (n_s) rather than the reduction of elementary CORDIC iterations. Previously, we have proposed a new method called *CORDIC angle recoding* for reducing the elementary iterations in forward CORDIC rotation mode [22], [23]. However, the more crucial case of backward rotation mode has not been addressed.

In the following sections, we will develop an algorithm to reduce the number of elementary rotations when a CORDIC processor is operated in the *backward* angle rotation mode. With very little hardware overhead, we show that the proposed algorithm will require fewer than $\lceil n/2 \rceil$ elementary rotations regardless which coordinate systems is used.

3 Backward CORDIC Angle Recoding

3.1 The Backward Angle Recoding Problem

Given $x(0)$, $y(0)$, for $k = 0, 1, 2, \ldots, k_{max}$, find $\mu(k) = 0$, or ± 1, and $i_k \in \{s(m, i); 0 \le i \le n-1\}$ via backward CORDIC rotations, such that

1) $|y(k_{max})| < \varepsilon$, where ε is the angle approximation error; and that
2) $k_{max} = \sum_{i=0}^{n-1}|\mu(i)|$ is minimized, where $\mu(i) = \pm 1$ if $i \in \{1 \le k \le k_{max}\}$, and $\mu(i) = 0$ otherwise.

Usually, ε is chosen to be equal to $2^{-s(m,n-1)}$ which, in terms, should be around 2^{-b} where b is the number of fraction digits. In the above formulation, k_{max} is the total number of elementary CORDIC iterations. In order to minimize k_{max}, as many elementary rotations need to be skipped as possible.

1. $\theta(0)$ is not needed in backward rotation mode, and sometimes is represented explicitly as the set of $\{\mu(i); 0 \le i \le n-1\}$ in forward rotation mode. In the latter case, (3) and (4) are also unnecessary.

This is a rather complicated problem since, in the worst case, there are $O(3^n)$ potential solutions. Instead of performing exhaustive search, in this paper, we propose a greedy heuristic which chooses i_k, $s(m, 0) \le i_k \le s(m, n-1)$, and $\mu(k) = sgn[x(k)] \cdot sgn[y(k)]$ in each CORDIC iteration such that

$$\left|y(k+1)\right| = \left|y(k) - \mu(k)x(k)2^{-i_k}\right| = \underset{0\le i\le n-1}{Min.}\left|y(k) - \mu(k)x(k)2^{-s(m,i)}\right|. \quad (7)$$

Within each iteration, in order to choose i_k, one needs to evaluate $2n-1$ additions (including $n-1$ comparisons) which is unacceptably expensive. Fortunately, since the sequence of the elementary rotation angles, $\{a_m(i); i = 0, n-1\}$, is monotonically decreasing, the same i_k which satisfies (7) can be found with only three additions. To see this, one observes that if there are two integers I_L and I_U, such that for $s(m, i) < I_L$,

$$y(k) - \mu(k)x(k)2^{-s(m,i)} \le y(k) - \mu(k)x(k)2^{-I_L} < 0,$$

and for $s(m, i) > I_U$,

$$y(k) - \mu(k)x(k)2^{-s(m,i)} \ge y(k) - \mu(k)x(k)2^{-I_U} > 0,$$

then the $s(m, i_k)$ which minimizes $|y(k) - \mu(k)x(k)2^{-s(m,i)}|$ must be in the range of $I_L \le s(m, i) \le I_U$. In order to find I_L and I_U, we need to express $|x(k)|$ and $|y(k)|$ in the normalized floating point representation such that

$$x(k) = sgn[x(k)]M_x(k)2^{E_x(k)}, \quad y(k) = sgn[y(k)]M_y(k)2^{E_y(k)}$$

where $\frac{1}{2} \le M_x(k), M_y(k) < 1$. Now we have the following result:

LEMMA 1. *Define an integer* $\delta_k = E_x(k) - E_y(k)$. *Then*

$$I_L = \delta_k - 1; \quad \textit{and } I_U = \delta_k + 1. \quad (8)$$

PROOF. With δ_k so defined, $x(k)$ is *aligned* with respect to $y(k)$:

$$y(k) - \mu(k)x(k)2^{-\delta_k} = sgn[y(k)]\left[M_y(k) - M_x(k)\right]2^{E_y(k)} \quad (9)$$

Now observe that $\frac{-1}{2} < \frac{-1}{2}M_x(k) \le \frac{-1}{4}$, and hence $0 < M_y(k) - \frac{1}{2}M_x(k) < \frac{3}{4}$. Therefore, if $y(k) > 0$, then $y(k) - \mu(k)x(k)2^{-\delta_k - 1} > 0$. Furthermore, this implies $y(k) - \mu(k)x(k)2^{-s(m,i)} > 0$ for $s(m, i) \ge \delta_k + 1$. On the other hand, if $y(k) < 0$, then $y(k) - \mu(k)x(k)2^{-\delta_k - 1} < 0$, and $y(k) - \mu(k)x(k)2^{-s(m,i)} < 0$ for $s(m, i) \ge \delta_k + 1$. Hence, one may set $I_U = \delta_k + 1$. Similarly, we have $-2 < -2M_x(k) \le -1$ and hence $\frac{-5}{2} < M_y(k) - 2M_x(k) < 0$. Therefore, for $s(m, i) \le \delta_k - 1$, if $y(k) > 0$, then $y(k) - \mu(k)x(k)2^{-s(m,i)} < 0$. Else, if $y(k) < 0$, then $y(k) - \mu(k)x(k)2^{-s(m,i)} > 0$. Hence, one may set $I_L = \delta_k - 1$. □

The result of Lemma 1 can be interpreted as follows: Compare M_y to $\frac{1}{2}M_x$, M_x, and $2M_x$. If $|M_y - \frac{1}{2}M_x| < |M_y - M_x|$, then choose $i_k = \delta_k - 1$. If, on the other hand, $|M_y - 2M_x| < |M_y - M_x|$, then choose $i_k = \delta_k + 1$. During this process, five binary additions need to be performed to evaluate i_k. Now, observe that $\frac{1}{2} \le M_x, M_y < 1$, it is clear that $\frac{1}{2}M_x \le M_y \le 2M_x$. Thus, the condition stated in Lemma 1 can be further simplified as follows:

LEMMA 2. *Denote* $\tau_1 \equiv 3 \cdot M_x - 2 \cdot M_y$, *and* $\tau_2 \equiv 3 \cdot M_x - 4 \cdot M_y$. *Then*

$$i_k = \begin{cases} \delta_k + 1, & \text{if } \tau_1 < 0; \\ \delta_k - 1, & \text{if } \tau_2 > 0; \\ \delta_k, & \textit{otherwise.} \end{cases} \quad (10)$$

PROOF. If $2M_x - M_y < M_y - M_x$, or equivalently, $3M_x - 2M_y < 0$, then $i_k = \delta_k + 1$. On the other hand, if $M_y - \frac{1}{2}M_x < M_x - M_y$, or equivalently, $3M_x - 4M_y > 0$, then $i_k = \delta_k - 1$. □

With Lemma 2, the number of additions need to be performed is reduced from five to three: One for evaluating $3M_x$, and the other two for evaluating τ_1 and τ_2.

The equations $\tau_1 = 0$ and $\tau_2 = 0$ represent two straight lines which intersect with the square region $\frac{1}{2} \le M_x, M_y \le 1$ in the $M_x - M_y$ plane. This is depicted in Fig. 1. These two straight lines divide this square region into three subregions, each with a different value assigned to i_k. Note that if $M_y > \frac{3}{4}$, i_k can only assume the value of $\delta_k + 1$ or δ_k, but not $\delta_k - 1$. On the other hand, if $M_y < \frac{3}{4}$, $i_k = \delta_k$, or $\delta_k - 1$, but not $\delta_k + 1$. That is to say, depending on whether M_y is greater or smaller than $\frac{3}{4}$ (which is easy to check), only τ_1 or τ_2 needs to be evaluated, but not both. This observation further reduces the number of additions to two. This result is proved in Theorem 1 below:

THEOREM 1. *With* i_k, δ_k, M_y, τ_1, *and* τ_2 *defined above, the* i_k *which satisfies* (7) *can be found as:*

$$i_k = \begin{cases} \delta_k + 1, & \text{if } M_y > \frac{3}{4} \text{ and } \tau_1 < 0; \\ \delta_k - 1, & \text{if } M_y < \frac{3}{4} \text{ and } \tau_2 > 0; \\ \delta_k & \textit{otherwise.} \end{cases} \quad (11)$$

PROOF. From Lemma 2, it is easy to verify that $M_y > \frac{3}{4}$ implies $\tau_2 = 3M_x - 4M_y < 0$. Thus only τ_1 needs to be computed. Similarly, $M_y < \frac{3}{4}$ implies $\tau_1 > 0$, thus only τ_2 needs to be computed. □

We now present the backward angle recoding algorithm (**BAR**) below:

CORDIC Backward Angle Recoding Algorithm (BAR):

Initialization: $[x(0)y(0)]^t$, $\{\mu(i) = 0; 0 \le i \le n-1\}$, $k = 0$, and ε.

While $|y(k)| > \varepsilon$, *Do*

1. Compute $i_k = \delta_k = E_x(k) - E_y(k)$, and $3M_x = M_x + 2M_x$.
2. *If* $M_y(k) \ge \frac{3}{4}$, then evaluate $\tau_1 = 3M_x - 2M_y$
   If $\tau_1 < 0$, then $i_k = \delta_k + 1$;
   Elseif $M_y(k) < \frac{3}{4}$, then evaluate $\tau_2 = 3M_x - 4M_y$
   If $\tau_2 > 0$, then $i_k = \delta_k - 1$;

3a. Elementary CORDIC rotation with Mantissa part only:

$$\begin{bmatrix} \tilde{M}_x(k+1) \\ \tilde{M}_y(k+1) \end{bmatrix} = \begin{bmatrix} 1 & m2^{-\delta_k - i_k} \\ -2^{\delta_k - i_k} & 1 \end{bmatrix} \begin{bmatrix} M_x(k) \\ M_y(k) \end{bmatrix} \quad (12)$$

3b. Normalize $\tilde{M}_x(k+1), \tilde{M}_y(k+1)$ and update $E_x(k+1)$, $E_y(k+1)$:

$$\begin{bmatrix} M_x(k+1) \\ M_y(k+1) \end{bmatrix} =$$

$$\begin{bmatrix} sgn\left[\tilde{M}_x(k+1)2^{\Delta_x}\right] & 0 \\ 0 & sgn\left[\tilde{M}_y(k+1)\right]2^{\Delta_y} \end{bmatrix}\begin{bmatrix}\tilde{M}_x(k+1)\\ \tilde{M}_y(k+1)\end{bmatrix}; \quad (13)$$

$$\begin{bmatrix}E_x(k+1)\\ E_y(k+1)\end{bmatrix} = \begin{bmatrix}E_x(k)\\ E_y(k)\end{bmatrix} + \begin{bmatrix}\Delta_x\\ \Delta_y\end{bmatrix}; \quad (14)$$

$$\begin{bmatrix}sgn[x(k+1)]\\ sgn[y(k+1)]\end{bmatrix} = \begin{bmatrix}sgn\left[\tilde{M}_x(k+1)\right]\cdot sgn[x(k)]\\ sgn\left[\tilde{M}_y(k+1)\right]\cdot sgn[y(k)]\end{bmatrix}. \quad (15)$$

4. $k = k + 1$.

End While_loop.

3.2 Discussion

1) Knowing that $i_k = \delta_k$ or $\delta_k \pm 1$, (12) can be simplified in actual implementation. However, i_k still needs to be computed explicitly because in the implementation of a systolic array of CORDIC processors [15], [16], [20], [21], backward rotation will be performed only in the head processor, and the remaining CORDIC processors will perform forward rotations based on $\mu(k)$ and i_k computed in the head processor.
2) Equation (13) can be realized with the usual mantissa normalization unit found in a normalized floating point arithmetic unit. It requires the detection of the number of leading zeros (or leading ones if the mantissa is in 2s complement representation, and is negative), and the decoding of it to a binary number.
3) Because the use of mantissa and exponent, it makes the BAR algorithm very suitable for normalized floating point CORDIC operations. We note that with fixed point to floating point format conversion, it will be equally useful for fixed point arithmetic as well.
4) Because $1/2 \le M_y < 1$, the convergence condition $|y(k)| < \varepsilon$ can be verified by an equivalent condition $b' + E_y > 0$ where $\varepsilon = 2^{-b'}$.

3.3 A Numerical Example

To illustrate the proposed BAR algorithm, let us consider the following numerical example: Suppose $x(0) = 0.10011010_2 \times 2^0$, and $y(0) = 0.11001101_2 \times 2^{-2}$, and $m = 1$ (circular rotation mode). Here we use 8 bits for mantissa and 4 bits for exponent (both excluding sign bits). Then, we have $\delta_0 = E_x(0) - E_y(0) = 0 - (-2) = 2$. Since $M_y(0) > 3/4$, we compute $\tau_1 = 3M_x - 2M_y = 0.00110100_2 > 0$. Hence $i_0 = \delta_0 = 2$. Next, according to (12), we compute

$$\tilde{M}_x(1) = M_x(0) + 2^{-2-2}M_y(0) = 0.10100110_2$$
$$\tilde{M}_y(1) = M_y(0) - 2^{2-2}M_x(0) = 0.00110011_2$$

For normalization, one has $\Delta_x = 0$, and $\Delta_y = -2$. This leads to new exponents $E_x(1) = 0$, and $E_y(1) = -4$. Also, $M_x(1) = \tilde{M}_x(1)$, $M_y(1) = 0.11001100_2$. One may verify that in the remaining of this iteration, $\tau_1 > 0$, and $M_x(2) = 0.10100111_2$, $M_y(2) = 0.10011000_2$. Also, $E_x(2) = 0$, and $E_y(2) = -6$. In this iteration, since $M_y(2) < 3/4$, we calculate and find τ_2, $= -.01101011_2$. Hence, we still have $\delta_2 = i_2$. Finally, since $\delta_3 = 6$, we have $\tilde{M}_y(3) = -0.00001111$, and $E_y(3) = 10$. If we choose $\varepsilon = 2^{-7}$, namely, $b' = -7$, then clearly the BAR algorithm will terminate at $k = 4$. Here we choose $b' = -7$ because there are only 8 significant bits in the mantissa, and hence $\tilde{M}_x$ will not be affected whenever $i_k + \delta_k \ge 2(E_x - E_y) - 1 > 8$. In fact, this is consistent with the recent quantization analysis of the CORDIC algorithm [17].

4 Convergence of the BAR Algorithm

The convergence of the BAR algorithm can be established by noting that $|y(k+1)| < |y(k)|$ after each iteration of the BAR algorithm. To ensure the condition $|y(k)| < \varepsilon$ is approximately equivalent to $|y(k)/x(k)| < \varepsilon$, we must constrain the magnitude of $|x(k)|$. However, this is readily accomplished in each iteration in the BAR algorithm during the normalization operation in step 3(b). Hence it is simple to reach the conclusion that the BAR algorithm will terminate in finite number of iterations. Below, we will derive a stronger result which bounds the total number of elementary CORDIC iterations needed using the BAR algorithm.

THEOREM 2. *If* $|\theta(0)| \le a_m(0)$, *using the backward angle recoding algorithm, then*

$$k_{\max} \le \lceil n/2 \rceil \quad (16)$$

PROOF. Let us consider the case of circular rotation ($m = 1$) first. During the kth iteration, according to (7), one chooses i_k such that, for any $i_k' \ne i_k$,

$$|y(k+1)| = |y(k) - \mu(i_k)x(k)\tan a_1(i_k)|$$
$$\le |y'(k+1)| = |y(k) - \mu(i_k')x(k)\tan a_1(i_k')|$$

where $\mu(i_k) = \mu(i_k') = \text{sign of } x(k)\cdot y(k)$. Equivalently, the above equation can be interpreted as to choose i_k such that for any $i_k' \ne i_k$,

$$\left|\tan|\theta_k| - \tan a_1(i_k)\right| \le \left|\tan|\theta_k| - \tan a_1(i_k')\right| \quad (17)$$

where $\tan\theta_k \equiv y(k)/x(k)$. Note that (17) *does not* imply that

$$\left||\theta_k| - a_1(i_k)\right| \le \left||\theta_k| - a_1(i_k')\right|$$

Our goal now is to show that, according to the BAR algorithm, if $i_k = i$, then either

$$i_{k+1} \ge i+2 \text{ or } i_{k+1} = i+1 \text{ and } i_{k+2} \ge i+4.$$

To see this, recall that according to (17), $i_k = i$ if and only if

$$\eta_{i-1} \ge |\theta_k| \ge \eta_i$$

where (using the fact that $\tan a_1(i) = 2^{-i}$)

$$\eta_i \equiv \tan^{-1}\left[\frac{\tan a_1(i) + \tan a_1(i+1)}{2}\right] = \tan^{-1}\left[3\cdot 2^{-i-2}\right] \ge \frac{1_1(i) + a_1(i+1)}{2}.$$

Hence,

$$|\theta_{k+1}| \equiv \left||\theta_k| - a_1(i_k)\right| \le \max.\{\eta_{i-1} - a_1(i), a_1(i) - \eta_i\} = \eta_{i-1} - a_1(i) \quad (18)$$

The last term in (18) is due to the fact that

$$\tan(\eta_{i-1} - a_1(i)) = \frac{3\cdot 2^{-i-1} - 2^{-i}}{1 + 3\cdot 2^{-i-1}\cdot 2^{-i}} = \frac{2^{-i-1}}{1 + 3\cdot 2^{-2i-1}}$$

$$\geq \tan(a_1(i) - \eta_i) = \frac{2^{-i} - 3\cdot 2^{-i-2}}{1 + 3\cdot 2^{-i-2}\cdot 2^{-i}}$$

$$= \frac{2^{-i-2}}{1 + 3\cdot 2^{-2i-2}}$$

for $i \geq 0$. Now, for $i \geq 2$, one has

$$\tan(\eta_{i-1} - a_1(i)) = \frac{2^{-i-1}}{1 + 3\cdot 2^{-2i-1}} > 3\cdot 2^{-i-3} = \tan\eta_{i+1}$$

Thus, it is possible that some $|\theta_{k+1}|$ will satisfy $|\theta_{k+1}| > \eta_{i+1}$. For these residue angles, the corresponding $i_{k+1} = i + 1$. When this occurs, our strategy is to show that the maximum possible value of the residue angle $|\theta_{k+2}|$, namely, $|(\eta_i - a_1(i)) - a_1(i+1)|$, must be smaller than η_{i+3}, and hence $i_{k+2} \geq i + 4$. To see this, first note that

$$\tan(a_1(i+1) - (\eta_i - a_1(i))) = \frac{2^{-i-1} - \dfrac{2^{-i-1}}{1 + 3\cdot 2^{-2i-1}}}{1 + 2^{-i-1}\cdot\dfrac{2^{-i-1}}{1 + 3\cdot 2^{-2i-1}}}$$

$$= \frac{3\cdot 2^{-3i-2}}{1 + 7\cdot 2^{-2i-2}}$$

Next, subtract $\tan\eta_{i+3}$ from the above:

$$\tan(a_1(i+1) - (\eta_i - a_1(i))) - \tan\eta_{i+3} =$$

$$\frac{3\cdot 2^{-3i-2}}{1 + 7\cdot 2^{-2i-2}} - 3\cdot 2^{-i-5} = 3\cdot 2^{-i-2}\left[\frac{2^{-2i}}{1 + 7\cdot 2^{-2i-2}} - 2^{-3}\right] < 0 \quad (19)$$

for $i \geq 2$. Equation (19) implies that $i_{k+2} > i + 4$, and hence completes the first part of the proof.

Next, we consider the hyperbolic rotation case ($m = -1$). Similar to the circular rotation case, one can show that the backward angle recoding algorithm is to choose i_k such that, for $i'_k \neq i_k$,

$$\left|\frac{|y(k)|}{|x(k)|} - \tanh a_{-1}(i_k)\right| \leq \left|\frac{|y(k)|}{|x(k)|} - \tanh a_{-1}(i'_k)\right|.$$

Define (note that $\tanh a_{-1}(i) = 2^{-i-1}$)

$$\eta_i \equiv \tanh^{-1}\left[\frac{\tanh a_{-1}(i) + \tanh a_{-1}(i+1)}{2}\right]$$

$$= \tanh^{-1}\left[3\cdot 2^{-i-3}\right] < \frac{a_{-1}(i) + a_{-1}(i+1)}{2}.$$

Then $i_k = i$ if and only if $\eta_i \leq |\theta_k| < \eta_{i-1}$. Clearly,

$$|\theta_{k+1}| \leq \max.\{a_{-1}(i) - \eta_i, \eta_{i-1} - a_{-1}(i)\} = \eta_{i-1} - a_{-1}(i). \quad (20)$$

This is because for $i \geq 0$,

$$\tanh[\eta_{i-1} - a_{-1}(i)] = \frac{3\cdot 2^{-i-2} - 2^{-i-1}}{1 - 3\cdot 2^{-i-2}\cdot 2^{-i-1}} = \frac{2^{-i-2}}{1 - 3\cdot 2^{-2i-3}}$$

$$> \tanh[a_{-1}(i) - \eta_i] = \frac{2^{-i-1} - 3\cdot 2^{-i-3}}{1 - 2^{-i-1}\cdot 3\cdot 2^{-i-3}} = \frac{2^{-i-3}}{1 - 3\cdot 2^{-2i-4}}. \quad (21)$$

Since

$$\tanh(\eta_{i-1} - a_{-1}(i) - a_{-1}(i+1)) = \frac{\dfrac{2^{-i-2}}{1 - 3\cdot 2^{-2i-3}} - 2^{-i-2}}{1 - 2^{-i-2}\cdot\dfrac{2^{-i-2}}{1 - 3\cdot 2^{-2i-3}}}$$

$$= \frac{3\cdot 2^{-3i-5}}{1 - 7\cdot 2^{-2i-4}} > 0$$

for $i > 0$, it is possible that $|\theta_{k+1}| > \eta_{i+1}$, and hence $i_{k+1} = i + 1$. Fortunately, compare the maximum residue angle for the $k + 2$ iteration, $\eta_{i-1} - a_{-1}(i) - a_{-1}(i+1)$, to η_{i+3}, one has

$$\tanh[\eta_{i-1} - a_{-1}(i) - a_{-1}(i+1)] - \tanh\eta_{i+3}$$

$$= \frac{3\cdot 2^{-3i-5}}{1 - 7\cdot 2^{-2i-4}} - 3\cdot 2^{-i-6}$$

$$= 3\cdot 2^{-i-6}\left[\frac{2^{-2i+1}}{1 - 7\cdot 2^{-2i-4}} - 1\right] < 0$$

for $i \geq 1$. Therefore, according to BAR, under the assumption that $i_k = i$, if $i_{k+1} = i + 1$ instead of $i + 2$, one must have $i_{k+2} \geq i + 4$. This concludes the proof for the hyperbolic rotation case.

Finally, we consider the linear rotation case ($m = 0$). In this case,

$$\lim_{m\to 0}\frac{1}{\sqrt{m}}\tan\sqrt{m}\,\theta_k = \theta_k = \frac{y(k)}{x(0)}.$$

Suppose that $a_0(i+1) \leq |\theta_k| \leq a_0(i)$, then according to the **BAR** algorithm,

$$|\theta_{k+1}| \equiv \frac{|y(k+1)|}{|x(0)|} = Min.\left\{\left|\frac{|y(k)|}{|x(0)|} - a_0(i)\right|, \left|\frac{|y(k)|}{|x(0)|} - a_0(i+1)\right|\right\}$$

$$\leq \frac{a_0(i) - a_0(i+1)}{2} = 2^{-i-2} = a_0(i+2).$$

Therefore, $|\theta_{k+1}|$ must skip the interval between $a_0(i+1)$ and $a_0(i+2)$. This concludes the proof for the linear rotation case, and hence this theorem. □

A simple simulation program in the MATLAB m-file format, has been developed to generate the distribution of iteration numbers when b = 8, 16, and 32. The results are depicted in Figs. 1a to 1c. For each simulation, 5,000 pairs of random data are generated in the range of [0, 1], and satisfying the conditions set in the Theorem 2. For all three different word lengths, not only the maximum number of iterations never exceeds $\lceil n/2 \rceil$ as predicted, we also observe that the average number of iterations is even lower (approximately $n/3$).

5 Improving the Usability of the BAR Algorithm

5.1 Extending Range of Convergence of Rotation Angles

Theorem 2 requires that the angle of initial coordinate $\theta(0) \leq a_m(0)$. This is often too restrictive for practical applications. In conventional CORDIC algorithm, the convergence range

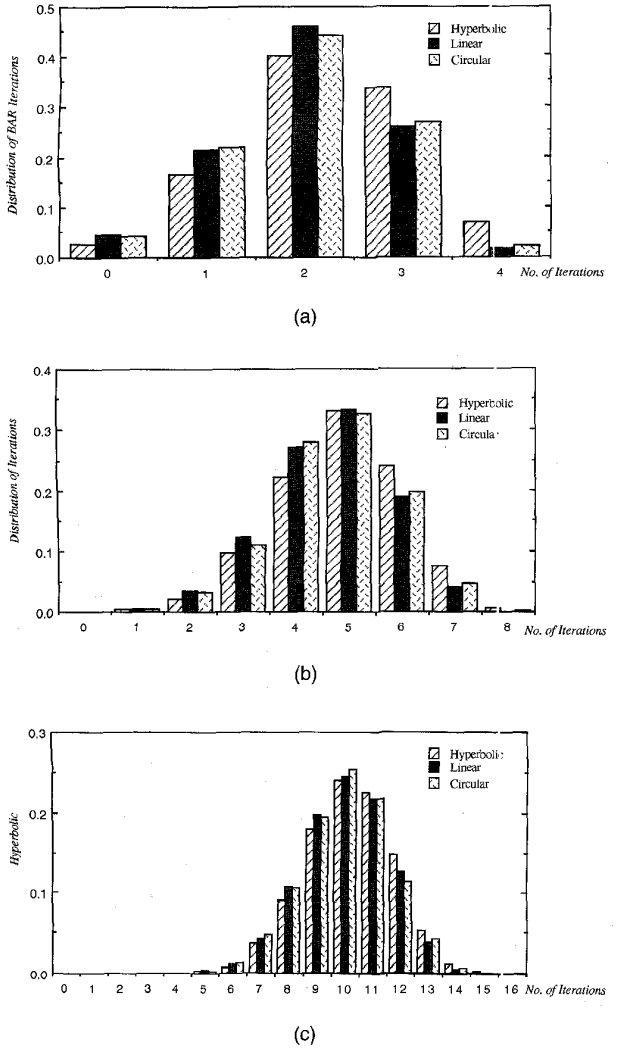


Fig. 1. Distributions of number of CORDIC iterations using BAR algorithm: (a) b = 8, (b) b = 16, (c) b = 32.

is chosen as the sum of all elementary rotation angles

$$\pm\sum_{i=0}^{n-1} a_m(i).$$

Even so, the range may still be insufficient in the case of hyperbolic rotations. One way to alleviate this problem is to use the "prescaling identities" as suggested by Walther [36]. Recently, X. Hu et al. [14] have proposed a method to extend the range of convergence using a simple, modified format of elementary angles. In this scheme,

$$a_{-1}(i) = \tanh^{-1}\left(1 - 2^{-2^{-i}}\right) = \frac{1}{2}ln\left(2^{2^{-i+1}} - 1\right)\Big] \quad \text{for } i = -1, \ldots, -I_o.$$

As such, $\tanh a_{-1}(i) = 1 - 2^{-2^{-i}}$. Clearly, $I_o \le \lfloor \log_2 b \rfloor$.

These extended elementary rotations in the hyperbolic rotation coordinate can be implemented with simple modification of the CORDIC architecture. First, one computes $(1 - 2^{-2^{-i}}) \cdot [x(i)y(i)]^t$. Then, $[x(i+1)y(i+1)]^t$ are evaluated using simple addition. Thus, with the addition of two parallel adders in each CORDIC processor, the number of range-extended elementary CORDIC iterations is bounded by I_o. It is tempting to try to extend the BAR algorithm to this set of elementary rotation angles with negative indices. Unfortunately, generalizing BAR to this set of extended elementary rotation angles does not guarantee reduction of the number of iterations, while adding considerably the complexity of hardware. Thus, we opt not to apply BAR to the range-extended iterations.

In this section, we will show that with these modified elementary rotation angles, the BAR algorithm is still applicable and will enjoy the benefit of saving 50% of elementary CORDIC iterations. The results are summarized in the following corollary:

COROLLARY 1. *Assuming that the scaling iterations are not to be performed, the total number of iterations $K_{\max}$ for each coordinate systems are:*

1) *If $m = 1$, and $x(0) \neq 0$, then $k_{\max} \le \lceil n/2 \rceil + 2$.*
2) *If $m = 0$, and $x(0) \neq 0$, then $k_{\max} \le \lceil n/2 \rceil + 1$.*
3) *If $m = -1$, $x(0) \neq 0$, and*

$$\left|\frac{y(0)}{x(0)}\right| < \tanh^{-1}\left[A_{\max}\right] = \tanh^{-1}\left[\sum_{i=I_o}^{n-1} \tanh^{-1} 2^{-i-1}\right],$$

then $k_{\max} \le \lceil n/2 \rceil + I_o$.

PROOF. 1) $m = 1$—If $x(0) > 0$, then $|\theta(0)| \le \pi/2$. Since $a_1(0) = \pi/4$, by choosing $i_1 = s(1, 0) = 0$, and $\mu(1) = sgn[y(0)]$,

$$|\theta(1)| = |\theta(0)| - a_1(0) \le \pi/4 = a_1(0)$$

Based on Theorem 2, it takes no more than $\lceil n/2 \rceil$ additional elementary CORDIC iterations. Hence, the total number of iterations is bounded by $\lceil n/2 \rceil + 1$. Now, if $x(0) < 0$, we may rotate the initial coordinate $[x(0)y(0)]^t$ by 180-degrees by simply changing the signs of both $x(0)$ and $y(0)$. If we count this sign-change as yet another elementary CORDIC iteration, then the upper bound $k_{\max}$ increases to $\lceil n/2 \rceil + 2$.

2) $m = 0$—If $|y(0)| \le |x(0)|$, then

$$|\theta(0)| = \lim_{m \to 0} \frac{1}{\sqrt{m}} \tan^{-1}\left[\sqrt{m}\left|\frac{y(0)}{x(0)}\right|\right] = \frac{|y(0)|}{|x(0)|} \le 1 = a_0(0).$$

Hence the result follows directly from Theorem 2. Now if $|y(0)| > |x(0)|$, we can always scale $y(0)$ such that $|y'(0)| = |y(0) \times 2^{-s}| \le |x(0)|$ ($s > 0$).This is equivalent to use an additional elementary rotation angle which equals to 2^s. Thus, the upper bound of the total number of iterations increases to $\lceil n/2 \rceil + 1$.

3) $m = -1$—The I_o iterations accounts for the I_o range-extended elementary iterations. After rotation through $a_{-1}(-1)$, the remaining θ_k will be less than $2a_{-1}(0)$. Thus, with one more iteration via $a_{-1}(0)$, the remaining $|\theta_k|$ will be less than $a_{-1}(0)$. Applying the results from Theorem 2, this part is proved. □

5.2 Discussion

With this corollary, the convergence range of the rotation angle is extended to practical limits. In the circular rotation case, any initial angle between 0 and 2π is covered. In the

linear rotation case, any number which is representable by the register length is applicable. In the hyperbolic rotation case, the hyperbolic rotation angle can be made very large so that $\tanh A_{\max} \to 1$. Two numerical details need to be considered during the implementation of this algorithm: First, when I_o becomes large, subtraction of two very close numbers may cause catastrophic loss of accuracy. Hence it is important to choose I_o appropriately. Another concern is the large magnitude of scaling factor associated with each negative-index iteration:

$$S_{-1}(i) = \frac{1}{\sqrt{1-\tanh^2(a(-i))}} = \frac{2^{2^{-i-1}}}{\sqrt{2-2^{-2^{-i}}}}, \quad -I_o \le i \le -1.$$

As such, $[x(i+1)y(i+1)]^t$ will need 2^{-i-1} more bits to represent. Hence some sort of preliminary scaling, such as multiplying by $2^{-2^{-i-1}}$ after each iteration will help reducing the explosion of dynamic ranges. When BAR is used, we propose to further normalize $|x(0)|$ to be between 0 and 1 to ensure the BAR iterations will converge. Here we assume that the iterations begins at $x(-I_o)$ and $y(-I_o)$, and the termination criterion is $|y(k)| < \varepsilon$.

With all these details implemented, we performed simulation to verify the results of Corollary 1. The results are depicted in Figs. 2a, 2b, and 2c.

5.3 Constant Scaling Factor BAR Algorithm

One impact of skipping a few elementary CORDIC iterations is that the scaling factor S_1 and S_{-1} will no longer be a constant as in the original CORDIC algorithm. This can easily be verified from (6): For circular rotation (m = 1), S_1 will vary between 1 and 0.6 (approximately). For m = –1, S_{-1} will vary between 1 and 1.2 (approximately). The results on $k_{\max}$ in Sections 3.1 and 3.2 are derived based on the assumption that the scaling operations can be completely by-passed. This is possible for a number of matrix based signal processing computing algorithms such as [18], [26], [29]. However, in other applications, a constant scaling factor is essential.

In this subsection, we will show that if the BAR algorithm is applied only to half of the fractional bits, the scaling factor will remain constant. This is a result reported by Lee and Lang [28]. This result is paraphrased in Theorem 3 in order to apply to the BAR algorithm.

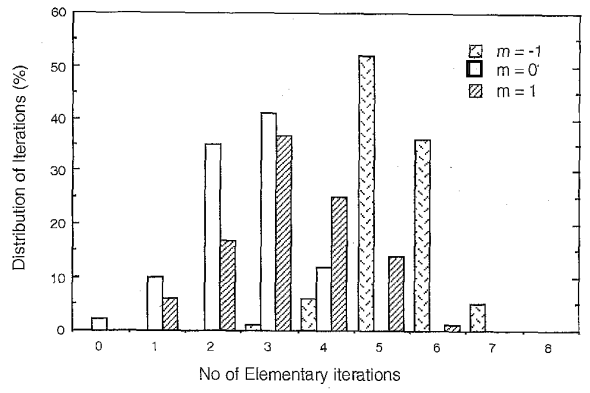
THEOREM 3. *If any elementary CORDIC iteration is skipped for $i > b/2$, where b is the number of fractional bits, the scaling factor will remain unchanged.*

PROOF. Denote S' to be the new scaling factor with all the elementary rotations skipped starting from $s(m, i) = s_1$ to $s(m, n-1) = b$. Then

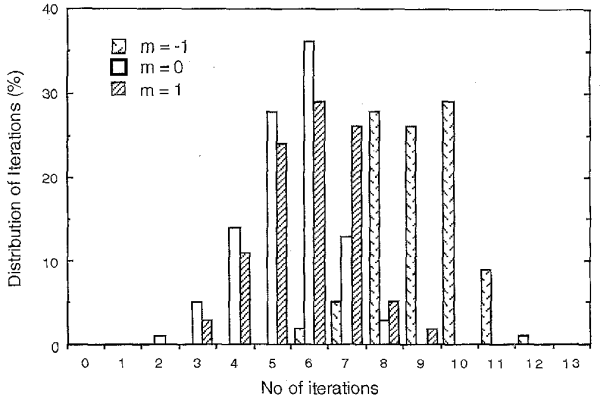
$$\frac{S'}{S} = \prod_{s=s_1}^{b} \sqrt{1+m\cdot 2^{-2s}} = \left[1 + m\sum_{s=s_1}^{b} 2^{-2s} + O\left(2^{-4s_1}\right)\right]$$

$$= \left[1 + m\cdot 2^{-2s_1}\cdot\frac{1-2^{-2(b-s_1+1)}}{1-2^{-2}} O\left(2^{-4s_1}\right)\right]$$

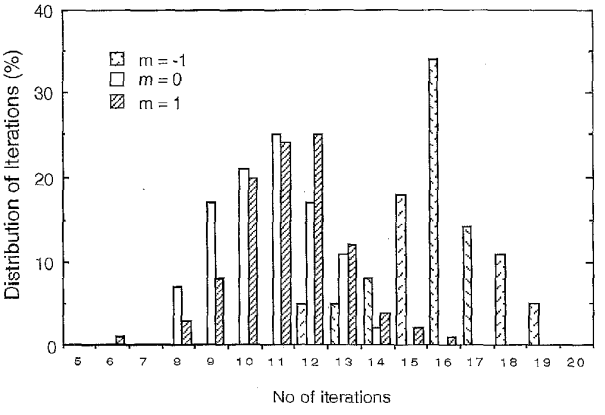
$$= 1 + m\cdot\frac{2}{3}\cdot 2^{-2s_1} + O\left(2^{-4s_1}\right)$$



(a)



(b)



(c)

Fig. 2. Distributions of number of CORDIC iterations using BAR algorithm with range of convergence extended: (a) b = 8, (b) b = 16, (c) b = 32.

where $O(2^{-4s_1})$ represents higher order terms whose magnitudes are smaller than 2^{-4s_1}. Clearly, if $s_1 \ge b/2 + 1$, $S'/S = 1$ accurate up to b fractional digits. □

The significance of Theorem 3 is that BAR can be applied to the last n/2 CORDIC elementary iterations without affecting the scaling factor. Using Theorem 2 and Corollary 1,

the total number of iterations, including both elementary rotation and scaling iteration becomes:

COROLLARY 2. *The total number of iterations, N_{total} using BAR and range extension method, while maintaining constant scaling coefficient are:*

1) *If $m = 1$, and $x(0) \neq 0$, then $N_{total} \leq \lceil 3n/4 \rceil + 2 + n_{s,1}$.*
2) *If $m = 0$, and $x(0) \neq 0$, then $N_{total} \leq \lceil 3n/4 \rceil + 1$.*
3) *If $m = -1$, $x(0) \neq 0$, and*

$$\left|\frac{y(-I_o)}{x(-I_o)}\right| < \tanh\left[A_{\max}\right] = \tanh\left[\sum_{i=-I_o}^{n-1} \tanh^{-1} 2^{-i-1}\right],$$

then $N_{total} \leq \lceil 3n/4 \rceil + 2I_o + n_{s,-1}$.

where $n_{s,1}$ and $n_{s,-1}$ are, respectively, the number of scaling iterations needed to multiply the constant scaling factor S_1, and S_{-1}.

This corollary follows directly from above discussion. Hence the proof is omitted in the interests of brevity.

5.4 Constant Iteration Numbers

The last modification we will discuss concerns the number of iterations. In a pipelined architecture, it is important to keep the number of pipeline stages constant to harness maximum parallelism. However, Theorem 2.2 only gives an upper bound. In fact, it is known that BAR will not yield constant number of iterations for each pair of vectors $[x(0)y(0)]^t$. However, since the number of iterations is upper bounded, if the actual number of iterations is less than the upper bound, one may simply pass the result through remaining pipeline stages without additional processing. Hence using the BAR algorithm will not affect the feasibility of a pipelined implementation.

6 CONCLUSION

In this paper, we proposed a *Backward Angle Recoding* method to reduce the number of CORDIC iterations. We proved that this method is capable of reducing more than 50% of the elementary CORDIC rotations. Integrating with existing CORDIC processor design methods, such as range extension, and constant scaling factor, we demonstrate the BAR algorithm is feasible for state-of-the-art CORDIC architecture.

REFERENCES

[1] H.M. Ahmed, "Signal Processing Algorithms and Architectures," PhD dissertation, Dept. of Electrical Eng., Stanford Univ., 1982.

[2] H.M. Ahmed and K.H. Fu, "A VLSI Array CORDIC Architecture," *Proc. ICASSP '89*, pp. 2,385-2,388, Glasgow, Scotland, 1989.

[3] J.D. Bruguera, E. Antelo, and E.L. Zapata, "Design of a Pipelined Radix-4 CORDIC Processor," *J. Parallel Computing*, vol. 19, no. 7, pp. 729-744, 1993.

[4] J.R. Cavallero and F.R. Luk, "CORDIC Arithmetic for an SVD Processor," *J. Parallel and Distributed Computing*, no. 5, pp. 271-290, 1988.

[5] T.W. Curtis, P. Allison, and J.A. Howard, "A CORDIC Processor for Laser Trimming," *IEEE Micro*, vol. 6, no. 3, pp. 61-71, 1986.

[6] A.A.J. De Lange, A.J. Vander Hoeven, E.F. Deprettere, and J. Bu, "An Optimal Floating Point Pipeline CMOS CORDIC Processor," *Proc. ISCAS*, Espoo, Finland, 1988.

[7] A.A.J. De Lange, E.F. Deprettere, A.J. Vander Veen, and J. Bu, "Real Time Applications of the Floating Point Pipeline CORDIC Processor in Massive-Parallel Pipelined DSP Algorithms," *Proc. ICASSP*, pp. 1,013-1,016, 1990.

[8] J.M. Delosme, "The Matrix Exponential Approach to Elementary Operations," *Proc. SPIE*, vol. 696, pp. 188-195, 1987.

[9] E.F. Deprettere, "Synthesis and Fixed Point Implementation of Pipelined True Orthogonal Filters," *Proc. ICASSP*, pp. 217-220, Boston, 1983.

[10] P. Dewilde, E.F. Deprettere, and R. Nouta, "Parallel and Pipelined VLSI Implementation of Signal Processing Algorithms," *VLSI and Modern Signal Processing*, S.Y. Kung et al., eds. Prentice Hall, 1985.

[11] R.G. Harber, X. Hu, J. Li, and S.C. Bass, "The Application of Bit-Serial CORDIC Computational Units to the Design of Inverse Kinematics Processors," *Proc. IEEE Int'l Conf. Robotics and Automation*, Philadelphia, 1988.

[12] G.L. Haviland and A.A. Tuszynski, "A CORDIC Arithmetic Processor Chip," *IEEE Trans. Computers*, vol. 29, no. 2, pp. 68-79, Feb. 1980.

[13] J.P. Hayes, *Computer Architecture and Organization*. McGraw-Hill, 1979.

[14] X. Hu, R.G. Harber, and S.C. Bass, "Expanding the Range of Convergence of the CORDIC Algorithm," *IEEE Trans. Computers*, vol. 40, no. 1, pp. 13-21, Jan. 1991.

[15] Y.H. Hu, "Pipelined CORDIC Architecture for the Implementation of Rotation-Based Algorithms," *Proc. Int'l Symp. VLSI Technology, Systems, and Algorithms*, Taipei, Taiwan, Republic of China, 1985.

[16] Y.H. Hu and S.Y. Kung, "Toeplitz Eigen-System Solver," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 33, no. 5, pp. 1,264-1,271, 1985.

[17] Y.H. Hu, "The Quantization Effects in the VLSI CORDIC Processor," *IEEE Trans. Signal Processing*, vol. 40, no. 4, pp. 834-844, Apr. 1992.

[18] Y.H. Hu and P.H. Milenkovic, "A Fast Least Square Deconvolution Algorithm for Vocal Tract Cross-Section Estimation," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 38, no. 6, pp. 921-924, 1990.

[19] Y.H. Hu and S. Naganathan, "A Novel Implementation of Chirp-Z Transform Using a CORDIC Processor," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 38, no. 2, pp. 3532-354, 1990.

[20] Y.H. Hu and H.M. Chern, "VLSI CORDIC Array Structure Implementation of Toeplitz Eigensystem Solvers," *Proc. ICASSP*, pp. 1,575-1,578, 1990.

[21] Y.H. Hu, "CORDIC-Based VLSI Architecture for Digital Signal Processing," *IEEE Signal Processing*, vol. 9, no. 3, pp. 16-35, July 1992.

[22] Y.H. Hu and S. Naganathan, "An Angle Recoding Method for CORDIC Algorithm Implementation," *IEEE Trans. Computers*, vol. 42, no. 1, pp. 99-102, Jan. 1993.

[23] Y.H. Hu, "A Forward Angle Recoding CORDIC Algorithm and Pipelined Processor Array Structure for Digital Signal Processing," *Digital Signal Processing P A Review J.*, vol. 3, no. 1, pp. 2-15, 1993.

[24] Y.H. Hu and Z. Wu, "An Efficient CORDIC Array Structure for the Implementation of Discrete Cosine Transform," *IEEE Trans. Signal Processing*, vol. 43, no. 1, pp. 331-336, 1995.

[25] K. Hwang, *Computer Arithmetic Principles, Architecture, and Design*. New York: John Wiley & Sons, 1979.

[26] K. Jainandunsing and E.F. Deprettere, "A New Class of Parallel Algorithm for Solving Systems of Linear Equation," *SIAM J. Science and Statistical Computing*, vol. 10, no. 5, pp. 880-912, 1989.

[27] D.T.L. Lee and M. Morf, "Generalized CORDIC for Digital Signal Processing," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, Paris, 1982.

[28] J.A. Lee and T. Lang, "Constant Factor Redundant CORDIC for Angle Calculation and Rotation," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 1,016-1,025, Aug. 1992.

[29] C.M. Rader, D.L. Allen, D.B. Glasco, and C.E. Woodward, "MUSE P A Systolic Array for Adaptive Nulling with 64 Degrees of Freedom, Using Givens Transformations and Wafer Scale Integration," MIT Lincoln Laboratory, May 18, 1990.

[30] T.Y. Sung, Y.H. Hu, and H.J. Yu, "Doubly Pipelined CORDIC Array Processor for Solving Toeplitz Systems," *Proc. 23rd Allerton Conf. Comm., Control, and Computing*, Monticello, Ill., 1985.

[31] T.Y. Sung, T. Parng, Y. Hu, and H.J. Yu, "Design and Implementation of a VLSI CORDIC Processor," *Proc. Int'l Symp. Circuits and Systems*, pp. 934-935, San Jose, Calif., 1986.
[32] T.Y. Sung, Y.H. Hu, and H.J. Yu, "Doubly Pipelined CORDIC Array for Digital Signal Processing Algorithms," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, pp. 69-72, Tokyo, 1986.
[33] D. Timmerman, H. Hahn, B.J. Hosicka, and G. Schmidt, "A Programmable CORDIC Chip for Digital Signal Processing Applications," *IEEE J. Solid-State Circuits*, vol. 26, no. 9, pp. 1,317-1,321, 1991.
[34] R. Udo, E. Deprettere, and P. Dewilde, "On the Implementation of Orthogonal and Orthogonalizing Algorithms Using Pipelined CORDIC Architectures," *Proc. Second European Signal Processing Conf.*, Erlangen, Germany, 1983.
[35] J.E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. Electronic Computers*, vol. 8, no. 3, pp. 330-334, 1959.
[36] J.S. Walther, "A Unified Algorithms for Elementary Functions," *Proc. Spring Joint Computer Conf.*, pp. 379-385, 1971.
[37] A.K. Yuen, "Intel's Floating-Point Processors," *Proc. Electron '88*, Boston, 1988.

Yu Hen Hu received the BSEE degree from National Taiwan University, Taipei, Taiwan, Republic of China, in 1976. He received the MSEE and PhD degrees in electrical engineering from University of Southern California, Los Angeles, in 1980 and 1982, respectively. From 1983 to 1987, he was an assistant professor in the Electrical Engineering Department of Southern Methodist University, Dallas, Texas. He joined the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, in 1987 as an assistant professor (1987-1989), and is currently an associate professor. His research interests include VLSI signal processing, artificial neural networks, spectrum estimation, fast algorithms and parallel computer architectures, and computer aided design tools for VLSI using artificial intelligence. He has published more than 100 journal and conference papers in these areas.

He is a former associate editor (1988-1990) for *IEEE Transaction of Acoustic, Speech, and Signal Processing* in the areas of system identification and fast algorithms. He is a founding member of the neural network signal processing technical committee and VLSI signal processing technical committee of the signal processing society.

Dr. Hu is a senior member of IEEE and a member of SIAM.

Homer H.M. Chern received the BS degree in electrical engineering from Chung-Cheng Institute of Technology, Taiwan, in 1972, an MS degree in computer science from Northwestern University, Evanston, Illinois, in 1981, and the PhD degree in electrical engineering from Southern Methodist University, Dallas, Texas, in 1990.

Since 1973, he has been a research assistant in the Electronic Division of Chung-Shan Institute of Science and Technology, where is presently an associate scientist in the System Development Division. His research interests include computer architecture, parallel processing, integration of inference functions into database machines, digital VLSI design, and VLSI signal processing. Dr. Chern's recent work concerns modeling, simulation, and gaming in the military wargame system in virtual environments by using concurrent engineering, spiral development, and usability engineering.