Constant-Factor Redundant CORDIC for Angle Calculation and Rotation

Jeong-A Lee, *Member, IEEE*, and Tomás Lang

***Abstract*— We develop a *Constant-Factor Redundant-CORDIC (CFR-CORDIC)* scheme, where the scale factor is forced to be constant while computing an angle for plane rotations. The direction of rotation is determined from an estimate of the sign and convergence is assured by suitably placed correcting iterations. Moreover, the number of iterations in the CORDIC rotation unit is reduced by about 25% by expressing the direction of the rotation in radix-2 and radix-4, and conversion to conventional representation is done on-the-fly. We estimate the performance of CFR-CORDIC and compare it with previously proposed schemes and show that it provides a similar execution time as redundant CORDIC with a variable scaling factor, with a significant saving in area.**

***Index Terms*— Angle calculation and rotation, constant scale factor, CORDIC, digital signal processing, matrix computations, matrix triangularization, redundant arithmetic.**

I. Introduction

MODERN digital signal processing requires the solution of systems of linear equations and the computation of eigenvalues, eigenvectors, singular values, and singular vectors [1]. Basic algorithms for these computations are matrix triangularization and singular value decomposition [2]. Since these applications are computationally intensive and require real-time response, parallel algorithms and pipelined and parallel architectures have been proposed to achieve high throughput [3]–[7]. In particular, linear, triangular, and mesh-connected arrays are very suitable because these matrix algorithms can be effectively mapped onto these arrays [8], [3], [5]. The key operations of these parallel algorithms are the computation of 2×2 rotation matrices (rotation angles) and their application to appropriate submatrices. The rotation angle is computed in boundary or diagonal processors and broadcast to other processors where the rotations are performed. One calculation approach is to compute the sine and cosine of the angle (by means of a sequence of operations involving squaring, addition, multiplication, square root, and division) and then to perform the rotations by multiplications and additions. The main drawback of this approach is that the angle calculation requires a long sequence of operations resulting in a large number of modules and a long execution time.

Another approach is to use the CORDIC algorithm [9], [10], which is utilized directly both to calculate the angle and to perform the rotations. It is characterized by its simple and regular architecture, which mainly consists of shifters and adders and is, therefore, well suited for VLSI implementation [11]–[13].

The CORDIC algorithm is relatively slow because each iteration requires an addition. A solution is to use a redundant number representation (for example, carry-save or signed-digit) to achieve carry-free addition. This can be applied directly to the rotation mode; however, for the angle-calculation mode it is necessary to detect the sign of the redundant representation, which is slow. In a sequential implementation of CORDIC, this sign detection can be done in parallel with the variable shifting. On the other hand, in an unfolded implementation the variable shifters are replaced by wired connections, resulting in lower delay (and a higher throughput if the implementation is pipelined). In such a case, the sign detection is in the critical path and degrades the performance. In [14] a scheme was developed, similar to what has been standard for other recurrences such as division and square root, in which an estimate of the sign is used. This approach produces a significant speedup for matrix computations but complicates the scaling required for rotation.

In this paper we develop a *Constant-Factor Redundant-CORDIC* (CFR-CORDIC) scheme for the angle calculation and rotation application. The scale factor is forced to be constant while computing the angles, so that the scaling for the associated rotations is performed as in the nonredundant CORDIC case. The approach, which is based on sign estimation and additional iterations to assure convergence, is an extension of that proposed in [15] for the calculation of sine and cosine. The angle calculation and rotation application is more complicated because the sign estimation is affected by the interdependence of the two CORDIC recurrence equations and because of the scaling required in the rotation.

We also propose a scheme to reduce by about 25% the number of iterations in the CORDIC rotation unit. This is achieved by expressing the direction of rotation in radix-2 and radix-4 (this can be used in both nonredundant and redundant CORDIC). Moreover, the transformation of the rotated output into nonredundant representation is done on-the-fly during the iterations, using an extension of the approach presented

Manuscript received November 1, 1991; revised February 18, 1992. This work was done while the authors were at the Computer Science Department, U.C.L.A., and was supported in part by the NSF Grant MIP-8813340 *Composite Operations Using On-Line Arithmetic for Application-Specific Parallel Architectures: Algorithms, Design, and Experimental Studies*. Based on "Matrix Triangularization by Fixed-Point Redundant CORDIC with a Constant Scale Factor" by J. Lee and T. Lang which appeared in Proceedings of SPIE Conference on Advanced Signal Processing Algorithms, Architectures, and Implementation, vol. 1348, pp. 430–447, San Diego, CA, July 1990.

J. Lee is with the Department of Electrical Engineering, University of Houston, Houston, TX 77204-4793.

T. Lang is with the Computer Architecture Department, Universitat Politecnica de Catalunya, Barcelona, Spain.

IEEE Log Numbver 9201909.

in [16]. The method here is more complex because of the iteration structure of the rotation. Finally, we discuss tradeoffs in the implementation, evaluate an estimate of performance and compare with previously proposed schemes. For matrix triangularization, we estimate that the execution time is similar to that of redundant CORDIC with variable scale factor, at significant savings in area.

II. Review of CORDIC Schemes

We briefly review the conventional (nonredundant) and redundant CORDIC schemes [10], [14] to compute an angle and perform rotations.

A. Nonredundant CORDIC

Angle calculation(vectoring mode): The angle $\theta = \tan^{-1}(Y_a[0]/X_a[0])$ in n-bit precision is obtained from $Z_a[n]$ using the following recurrence equations with $Z_a[0] = 0$.

$$\begin{aligned} X_a[i+1] &= X_a[i] + \sigma_i 2^{-i} Y_a[i] \\ Y_a[i+1] &= Y_a[i] - \sigma_i 2^{-i} X_a[i] \\ Z_a[i+1] &= Z_a[i] + \sigma_i \tan^{-1}(2^{-i}) \end{aligned} \tag{1}$$

where $X_a[0]$ is assumed positive and the direction of the rotation is obtained as

$$\sigma_i = \begin{cases} +1 & \text{if } Y_a[i] \geq 0 \\ -1 & \text{if } Y_a[i] < 0. \end{cases} \tag{2}$$

Rotation(rotating-mode): To perform a plane rotation on the input vector $(X_r[0], Y_r[0])$ by an angle $(\theta = Z_r[0])$, the following recurrence equations are used:

$$\begin{aligned} X_r[i+1] &= X_r[i] + \sigma_i 2^{-i} Y_r[i] \\ Y_r[i+1] &= Y_r[i] - \sigma_i 2^{-i} X_r[i] \\ Z_r[i+1] &= Z_r[i] + \sigma_i \tan^{-1}(2^{-i}) \end{aligned} \tag{3}$$

where

$$\sigma_i = \begin{cases} -1 & \text{if } Z_r[i] \geq 0 \\ +1 & \text{if } Z_r[i] < 0. \end{cases} \tag{4}$$

The recurrence equations are the same for both angle and rotation modes.

Rotation with angle in decomposed form: Note that for an application in which an angle is calculated by the vectoring mode and then this angle is used for rotation, it is not necessary to implement the Z recurrences in any of the two modes, since the angle can be used in its σ-decomposed form. This is the type of application we have in mind, so that we do not include the Z recurrences in the sequel and the σ for rotation is obtained from the angle calculation. The general architecture of these applications is shown in Fig. 1.

In both operation modes, the iteration time corresponds to the delay of the variable shifter plus the delay of the adder. If the recurrences are unfolded, the variable shifter is replaced by wiring, which reduces the iteration time to the delay of the adder. Moreover, the unfolded implementation can be pipelined to increase the throughput.

Scaling: Since the CORDIC rotation changes the length of the vector, to maintain the same length of the input vector

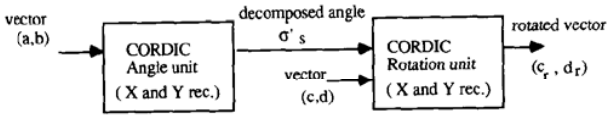

Fig. 1. General architecture for matrix computations with CORDIC.

$(X_r[0], Y_r[0])$, the following CORDIC scaling operation is necessary, where (x^R, y^R) is the rotated vector.

$$\begin{bmatrix} x^R \\ y^R \end{bmatrix} = \frac{1}{K} \begin{bmatrix} X_r[n] \\ Y_r[n] \end{bmatrix} \text{ where}$$
$$K = \prod_{i=0}^{n-1} 1/\cos(\sigma_i 2^{-i}) = \prod_{i=0}^{n-1} \sqrt{1 + \sigma_i^2 2^{-2i}}. \tag{5}$$

Note that K is *constant* for nonredundant CORDIC since $\sigma_i^2 = 1$ for all i.

The scaling operation can be done by multiplication by $1/K$. However, as proposed in [17]–[19] it might be simpler to include scaling iterations of the form $X_r[i] \leftarrow X_r[i] \pm 2^{-j} X_r[i]$ and $Y_r[i] \leftarrow Y_r[i] \pm 2^{-j} Y_r[i]$ and repetitions of CORDIC iterations to make the scaling factor equal to a power of two (really 1 or 2) and perform the scaling by shifting. For an efficient implementation, the total number of these additional iterations need to be minimized; solutions have been found that require about 25% more iterations to force K to be 2 or 1 [19], [20].

B. Redundant CORDIC

To reduce the iteration time, it is possible to use a redundant representation (for example carry-save or signed-digit) and the corresponding (carry-free) adder. In the development that follows we consider the signed-digit case; the carry-save alternative is similar. We now discuss the effect that the use of the redundant representation has on the angle computation and on the rotation.

Angle computation: To compute σ from the redundant representation, a sign detection is required, which is slow; however, in a sequential (not unfolded) implementation this sign detection is not in the critical path since it can be performed in parallel with the variable shifter. On the other hand, for an unfolded implementation, since the shifters are eliminated, the delay of this sign detection determines the iteration time. In [14] a scheme was proposed that computes $\hat{\sigma}_i$ based on an estimate of $Y[i]$. This requires the modification of the recurrence and selection function to determine the direction of rotation. By making $W[i] = 2^i Y_a[i]$ we get from (1),

$$\begin{aligned} X_a[i+1] &= X_a[i] + \hat{\sigma}_i 2^{-2i} W[i] \\ W[i+1] &= 2(W[i] - \hat{\sigma}_i X_a[i]). \end{aligned} \tag{6}$$

The value of $\hat{\sigma}_i$ is obtained from $\hat{W}[i]$, an estimate produced by truncating $W[i]$ to one fractional (signed) bit. It is shown in [14] that to assure convergence the value of $\hat{\sigma}_i$ has to belong

to the set $\{-1, 0, 1\}$ and

$$\hat{\sigma}_i = \begin{cases} 1 & \text{if } \hat{W}[i] \geq \frac{1}{2} \\ 0 & \text{if } \hat{W}[i] = 0 \\ -1 & \text{if } \hat{W}[i] \leq -\frac{1}{2} \end{cases} \quad (7)$$

where $X_a[0]$ and $Y_a[0]$ ($W[0]$) are fractional values, one of them is normalized, and $X_a[0] > 0$. With these conditions, a normalized $X_a[1]$ is obtained, that is, $1/2 \leq X_a[1] < 2$.

Rotation: Redundant adders can be used for the iterations. No $\hat{\sigma}$ is calculated since it is obtained from the angle calculation.

Scaling: The same scaling operation for rotation as in nonredundant CORDIC has to be performed. However, in this redundant CORDIC approach, the scale factor K is *not a constant* as $\hat{\sigma}_i \in \{-1, 0, 1\}$. Therefore, K must be computed for each $\hat{\sigma}$-decomposed angle. This calculation can be performed in two steps [14]: 1) K^2 computation using $P[j+1] = P[j] + |\hat{\sigma}_j| 2^{-2j} P[j]$ with initial condition $P[0] = 1$ and 2) $K = \sqrt{P}$. The K computation can be overlapped with the angle computation, so it does not increase the computation time. However, a division operation is needed for the final scaling operation.

It was estimated that this scheme produces a speedup of 4.5 for matrix triangularization and of 4 for SVD, when compared with the nonredundant scheme [14].

III. Constant-Factor Redundant CORDIC

To reduce the implementation cost of redundant CORDIC, especially for the scale factor calculation and the scaling operation, there is a need to develop a redundant CORDIC scheme with a constant scale factor. This is achieved if $\hat{\sigma}$ is restricted to the set $\{-1, 1\}$. However, since to eliminate the delay of exact sign detection, $\hat{\sigma}$ is obtained from an estimate of $Y_a[i]$, this does not assure convergence. We now discuss how to modify the scheme to assure convergence, first in the calculation of the sine and cosine functions (a review of a scheme proposed in [15]) and then in angle calculation/rotation mode (a new scheme).

A. Sine and Cosine Calculations

In [15] a *correcting iteration scheme* is used to have a constant scale factor when redundant CORDIC computes the cosine and sine functions. For these computations, the recurrences (3) are used and $\hat{\sigma}_i$ is obtained as in (4), except that an estimate of $Z[i]$ is utilized. Because of this estimate, to assure convergence it is necessary to repeat some iterations, called *correcting iterations*. These repetitions are done at fixed intervals, where the frequency of repetition depends on the precision of the estimate.

This approach could be directly extended to perform rotations by decomposing the given angle. However, it is not directly applicable when the angle is computed as $\tan^{-1}(a/b)$ and this angle is used in decomposed form for the rotations. We develop the required modifications now.

B. Angle Calculation/Rotation

In the case of angle calculation, the problem of having a constant scale factor is more complicated because the direction of the rotation is obtained from an estimate of $Y_a[i]$ and it is necessary to deal with the inter-dependency of the recurrences of X and Y. To use the estimate effectively, we utilize the modified recurrences of expressions (6). However, to achieve a constant scaling factor, instead of the $\hat{\sigma}$-selection of expression (7), we use

$$\hat{\sigma}_i = \begin{cases} 1 & \text{if } \hat{W[i]} \geq 0 \\ -1 & \text{if } \hat{W[i]} < 0 \end{cases} \quad (8)$$

where $\hat{W[i]}$ is an estimate obtained by truncating the signed-digit representation of $W[i]$ to the tth fractional bit. As before, this requires $X_a[1] \geq 1/2$, which is achieved in the same manner.

The use of this $\hat{\sigma}$ selection does not assure convergence because $W[i]$ can attain values outside the bound required for convergence. We now compute the convergence bound, determine the amount by which $W[i]$ can surpass this bound because of the $\hat{\sigma}$-selection used, and show how correcting iterations can restore the bound.

Theorem 3.1 (Bound): When the direction of the angle σ_i is obtained using the exact sign, that is,

$$\sigma_i = \begin{cases} 1 & \text{if } W[i] \geq 0 \\ -1 & \text{if } W[i] < 0 \end{cases}$$

$W[i]$ is bounded as follows [14]:

$$|W[i]| < 2X[i].$$

Proof: Since σ_{i-1} is determined by the sign of $W[i-1]$, the largest magnitude of $W[i]$ is obtained when $W[i-1] = 0$ ($\sigma_{i-1} = 1$) in

$$W[i] = 2(W[i-1] - \sigma_{i-1} X[i-1]).$$

Thus,

$$|W[i]| < 2X[i-1].$$

As shown in [21], $X[i] \leq X[i+1]$ for all i. Thus,

$$|W[i]| < 2X[i-1] \leq 2X[i]. \quad \square$$

Fig. 2 shows the bound of Theorem 3.1. This bound cannot be maintained with the selection function of the CFR-CORDIC scheme [expression (8)], as shown by the following lemma.

Lewmma 3.1: Assume that $W[p]$ satisfies the bound of Theorem 3.1, that is $|W[p]| < 2X[p]$. If $\hat{\sigma}_p$ is determined from $\hat{W}[p]$ where $\hat{W}[p]$ is computed using t fractional bits of $W[p]$, then $W[p+1]$ satisfies the new bound

$$-2X[p+1] - 2 * 2^{-t}(1 + 2^{-2p}) < W[p+1] < 2X[p+1].$$

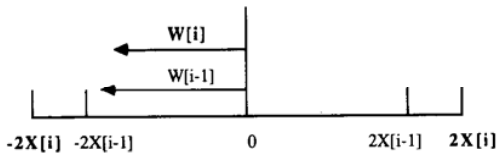

Fig. 2. Convergence bound.

Proof: Three regions of $\hat{W}[p]$ need to be considered.

Case a): $\hat{W}[p] > 0$ ($\hat{\sigma}p = 1$).

In signed-digit number system, if $\hat{W}[p] > 0$ then $W[p] > 0$, since $\hat{W}[p]$ is computed using t fractional bits of $W[p]$. Thus, $\hat{\sigma}_p = \sigma_p = 1$. From Theorem 3.1, $|W[p+1]| < 2X[p+1]$.

Case b): $\hat{W}[p] = 0$ ($\hat{\sigma}p = 1$).

If $\hat{W}[p] = 0$ then $-2^{-t} < W[p] < 2^{-t}$ We divide this region into two. In the first, when $-2^{-t} < W[p] < 0$, $\sigma_p = -1$ (not equal to $\hat{\sigma}$). From the recurrence,

$$W[p+1] = 2(W[p] - X[p])$$
$$X[p+1] = X[p] + 2^{-2p}W[p].$$

Consequently,

$$W[p+1] = 2W[p] - 2(X[p+1] - 2^{-2p}W[p])$$
$$= -2X[p+1] + 2W[p](1 + 2^{-2p})$$

and for the most negative value of $W[p]$

$$W[p+1] > -2X[p+1] - 2 * 2^{-t}(1 + 2^{-2p}).$$

On the other hand, when $0 < W[p] < 2^{-t}$ then $\sigma_p = \hat{\sigma}_p = 1$. Thus, from Theorem 3.1, $|W[p+1]| < 2X[p+1]$.

Case c): $\hat{W}[p] < 0$

Similar to case a), $\hat{\sigma}_p = \sigma_p = -1$. Thus, from the Theorem 3.1, $|W[p+1]| < 2X[p+1]$.

From a), b), and c), the proof is done. □

Fig. 3 shows the case when $W[p+1]$ is out of the bound due to the use of an estimate, $\hat{W}[p]$. Once $W[i]$ is outside of the bound required for convergence, the amount outside of the bound accumulates in the following iterations, as shown in the following lemma.

Lemma 3.2: If $-2X[p+1] - 2*2^{-t}(1+2^{-2p}) < W[p+1] < -2X[p+1]$, then the bound of iteration $W[p+k]$ where $k = \{2, 3, \cdots\}$ becomes

$$-2X[p+k] - 2^{k-t}(1+2^{-2p}) < W[p+k] < -2X[p+k].$$

Proof by induction:

1) Basis condition for $k = 2$,

Since $-2X[p+1] - 2 * 2^{-t}(1 + 2^{-2p}) < W[p+1] < -2X[p+1]$, we obtain $\hat{W}[p+1] < 0$. Thus, $\hat{\sigma}_{p+1} = -1$. From the lower bound and the recurrence, we have

$$W[p+2] > 2(-2X[p+1] - 2 * 2^{-t}(1+2^{-2p}) + X[p+1])$$
$$W[p+2] > 2(-X[p+2] - 2^{-2(p+1)}W[p+1] - 2 * 2^{-t}(1+2^{-2p}))$$
$$W[p+2] > -2X[p+2] - 2^{2-t}(1+2^{-2p}).$$

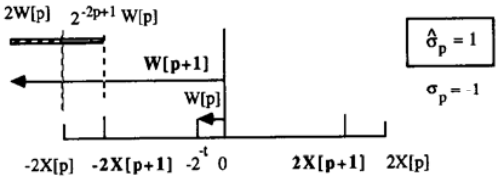

Fig. 3. Out of the bound due to the use of an estimate.

From the upper bound and the recurrence we have

$$W[p+2] < 2(-2X[p+1] + X[p+1]) = 2(-X[p+1])$$
$$W[p+2] < 2(-X[p+2] - 2^{-2(p+1)}W[p+1])$$
$$< -2X[p+2].$$

Thus,

$$-2X[p+2] - 2^{2-t}(1+2^{-2p}) < W[p+2] < -2X[p+2].$$

2) Induction step. Assume that it is true when $k = i$ and prove that it is true when $k = i+1$.

By assumption,

$$-2X[p+i] - 2^{i-t}(1+2^{-2p}) < W[p+i] < -2X[p+i].$$

Therefore, $\hat{W}[p+i] < 0$ and $\hat{\sigma}_{p+i} = -1$. From lower bound of assumption and from recurrence,

$$W[p+i+1] > 2(-2X[p+i] - 2^{i-t}(1+2^{-2p}) + X[p+i])$$
$$W[p+i+1] > 2(-X[p+i+1] - 2^{-2(p+i)}W[p+i] - 2^{i-t}(1+2^{-2p}))$$
$$W[p+i+1] > -2X[p+i+1] - 2^{i+1-t}(1+2^{-2p}).$$

From upper bound of assumption and from recurrence,

$$W[p+i+1] < 2(-2X[p+i] + X[p+i]) = 2(-X[p+i])$$
$$W[p+i+1] < 2(-X[p+i+1] - 2^{-2(p+i)}W[p+i])$$
$$< -2X[p+i+1].$$

Thus,

$$-2X[p+i+1] - 2^{i+1-t}(1+2^{-2p}) < W[p+i+1] < -2X[p+i+1].$$

Therefore, from 1) and 2), the proof is done. □

Fig. 4 shows the bound of $W[p+2]$ to explain how the amount outside of the bound accumulates once $W[p+1]$ is out of the bound.

To recover the original bound of $|W[i+1]| < 2X[i+1]$, a correcting iteration needs to be performed. The same angle of the previous iteration is used for the correcting iteration, i.e., 2^{-i} instead of $2^{-(i+1)}$ is used with $Y[i+1]$ as shown below.

$$X^C[i+1] = X[i+1] + \hat{\sigma}_i^C 2^{-i} Y[i+1]$$
$$Y^C[i+1] = Y[i+1] - \hat{\sigma}_i^C 2^{-i} X[i+1].$$

The corresponding shifted recurrence becomes

$$X^C[i+1] = X[i+1] + \hat{\sigma}_i^C 2^{-2i-1} W[i+1]$$
$$W^C[i+1] = W[i+1] - 2\hat{\sigma}_i^C X[i+1].$$

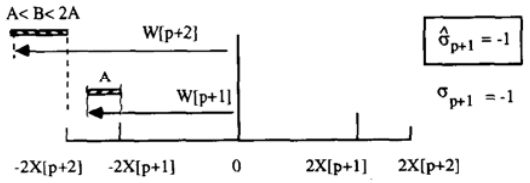

Fig. 4. Accumulating the amount outside of the bound.

The direction of the angle $\hat{\sigma}_i^C$ is determined from $\hat{W}[i+1]$ instead of $\hat{W}[i]$. However, notice that the same selection function in (8) is used for this iteration. Fig. 5 shows the case where a correcting iteration recovers the bound of Theorem 3.1. The next theorem determines the frequency of correcting iterations required.

Theorem 3.2: Assume that $|W[p]| < 2X[p]$ and the direction of the angle, $\hat{\sigma}_{p+k}$, is determined from the estimate $\hat{W}[p+k]$ for all k. If t fractional bits are used in the estimate, then a maximum of $t-1$ iterations can be done before a correcting iteration to recover the convergence bound.

Proof: Let us assume that k iterations are performed after the pth iteration. From Lemma 3.2, the bound of $W[p+k]$ is expanded as follows due to the use of estimate

$$-2X[p+k] - 2^{k-t}(1+2^{-2p}) < W[p+k] < 2X[p+k].$$

If $|W[p+k]| < 2X[p+k]$, then it satisfies the bound of Theorem 3.1 and a correcting iteration is not needed to recover the bound.

Since we are interested in finding out how many iterations can be performed before recovering the bound by a correcting iteration, we need to consider the case when $W[p+k]$ is in the following range:

$$-2X[p+k] - 2^{k-t}(1+2^{-2p}) < W[p+k] < -2X[p+k].$$

In this case, $\hat{\sigma}_{p+k} = -1$ since $\hat{W}[p+k] < 0$. By applying the correcting iteration, $W^C[p+k]$ becomes

$$W^C[p+k] = W[p+k] + 2X[p+k] > -2^{k-t}(1+2^{-2p}).$$

To recover the bound of Theorem 3.1, we need to find k satisfying the following inequality,

$$|-2^{k-t}(1+2^{-2p})| < 2X[p+k].$$

Therefore,

$$2^{k-t} < \frac{2X[p+k]}{(1+2^{-2p}).}$$

The value of k is determined by considering the smallest possible value of the right-hand side. For this, we consider the smallest value of $2X[p+k]$ and the largest of $(1+2^{-2p})$, i.e., we should consider the smallest value of p. Since we assume that the input $W[0]$ is in conventional number representation, the direction of the angle $\hat{\sigma}_0$ is correct, regardless of the use of an estimate. Thus, the smallest p to consider is 1. Now we want to determine the smallest value of $X[p+k]$. In [22] it is shown that the smallest value of $X[p+k]$ is $3/8$. Therefore,

$$k - t < \frac{\log \frac{2X[p+k]}{(1+2^{-2p})}}{\log 2} < \frac{\log \frac{3}{5}}{\log 2}.$$

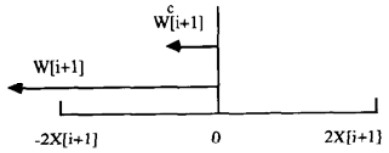

Fig. 5. Recovering the convergence bound by a correcting iteration.

Thus, the largest integer k becomes $t-1$. □

As indicated by Theorem 3.2, a correcting iteration puts W inside the convergence bound, if it was outside of the bound before this iteration. However, as shown in the next lemma, this correcting iteration might take W out of the bound if it was inside the bound before.

Lemma 3.3: If $|W[i+1]| < 2X[i+1]$, the bound after a correcting iteration

$$W^C[i+1] = W[i+1] - 2\hat{\sigma}_i^C X[i+1]$$

becomes

$$-2^{-t} - 2X[i+1] < W^C[i+1] < 2X[i+1].$$

Proof: The proof is very similar to that of Lemma 3.1 and is omitted. □

From Lemma 3.2, Theorem 3.2, and Lemma 3.3, we obtain the following corollary.

Corollary 3.1: If t fractional bits are used for the estimate, then the interval between correcting iterations should be less than or equal to $(t-2)$. □

The fact that a correcting iteration does not have to be included exactly every $(t-2)$ iterations provides flexibility and helps to minimize the number of additional iterations required to avoid an explicit scaling operation. Also, note that it is possible to repeat a correcting iteration more than once if it helps to minimize the total number of iterations for a certain K. This additional correcting iterations should not be counted as one of the iterations of CORDIC when determining the index of the next correcting iteration.

Since the frequency of correcting iterations is determined by t, to reduce their number it is necessary to increase t. However, this makes the σ selection slower. Consequently, the value of t has to be chosen to reduce the overall execution time.

C. Reducing the Number of Correcting Iterations

From the previous discussion we can determine the minimum number of correcting iterations required. We now show that it is sufficient to have only one correcting iteration in the second half, if for that part the $\hat{\sigma}$-selection of expression (7) is used. That is, we split the iterations of CORDIC (for $i=0$ to $i=n-1$) into two groups as follows:

1) Selection function for $0 \le i \le n/2$

$$\hat{\sigma}_i = \begin{cases} 1 & \text{if } \hat{W}[i] \ge 0 \\ -1 & \text{if } \hat{W}[i] < 0. \end{cases} \tag{9}$$

As shown earlier, correcting iterations must be included for convergence for this group and the number of fractional bits used for $\hat{W}[i]$ determines the frequency of correcting iterations.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.