# SVD by Constant-Factor-Redundant-CORDIC *

Jeong-A Lee

Department of Electrical Engineering
University of Houston
Houston, TX 77204-4793

Tomás Lang

Computer Architecture Department
Universitat Politecnica de Catalunya
Barcelona, Spain

## Abstract

*We develop a Constant-Factor-Redundant-CORDIC (CFR-CORDIC) scheme where the scale factor is forced to be constant while computing angles for SVD. Based on the scheme, we present a fixed-point implementation of SVD with the following additional features: (1) the final scaling operation is done by shifting, (2) the number of iterations in CORDIC rotation unit is reduced by about 25% by expressing the direction of the rotation in radix-2 and radix-4, and (3) the conventional number representation of rotated output is obtained on-the-fly, not from a carry-propagate adder. We compare this scheme with previously proposed ones and show that it provides similar execution time as redundant CORDIC with variable scaling factor with significant saving in area.*

## 1 Introduction

Singular value decomposition(SVD) is an important tool in a variety of applications of digital signal processing [1, 2]. Since the decomposition consists of computation-intensive tasks, parallel algorithms and architectures have been developed to get an adequate throughput rate (processing time) [3, 4].

Among the parallel algorithms and architectures for SVD, we consider the one presented by Brent, Luk, and Van Loan [3]. It consists of a mesh-connected processor array, where diagonal processors compute left and right angles to diagonalize a 2x2 matrix. These angles are passed to off-diagonal processors in the column and row directions to perform the two-sided plane rotations.

The angle calculation and the rotation can be performed using standard arithmetic modules such as adders, multipliers, dividers, and square root units [4]. However, in this case the angle calculations require a long sequence of operations resulting in a large number of modules and a long execution time. An attractive alternative presented by Cavallaro and Luk [5] is to use the CORDIC algorithm which is effective for both the angle calculations and the rotations. The execution time of this approach was reduced by the redundant-CORDIC scheme presented by Ercegovac and Lang [11], which has the drawback that the scaling factor is not constant and has to be calculated for each angle.

In this paper we develop an algorithm and an implementation for SVD using Constant-Factor-Redundant-CORDIC, which is based on the scheme developed by Takagi, Asada, and Yajima for the computation of sine and cosine [12].

We review previous work on CORDIC for SVD in Section 2. Following that, we develop a Constant-Factor-Redundant-CORDIC scheme, evaluate the performance of this new scheme with respect to time and space, and compare it with previously proposed schemes in Section 3. The summary and conclusion are presented in Section 4.

## 2 Previous CORDIC work for SVD

We briefly review the conventional(non-redundant) and redundant CORDIC schemes to compute angles for SVD and perform the corresponding rotations.

### 2.1 Non-redundant CORDIC

Cavallaro and Luk presented a CORDIC processor for SVD [5] using conventional(non-redundant) arithmetic based on the original CORDIC algorithm [6, 7]. The two modes of operation are as follows:

**Angle calculation(vectoring mode):** The angle $\theta = \tan^{-1}(\frac{Y_a[0]}{X_a[0]}) = Z_a[n]$ is computed from the following recurrence equations with $Z_a[0] = 0$.

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

where, the direction of the rotation $\sigma_i$ is obtained as

$$\sigma_i = \begin{cases} 1 & \text{if } Y_a[i] \geq 0 \\ -1 & \text{if } Y_a[i] < 0 \end{cases} \quad (2)$$

To compute $\theta$ with $n$-bit precision, $n$ CORDIC iterations need to be executed. The step time of one CORDIC iteration is determined by $\sigma$ computation,

*This research was done while the authors were at the Computer Science Department, UCLA, and was supported in part by the NSF Grant No. MIP-8813340 *Composite Operations Using On-Line Arithmetic for Application-Specific Parallel Architectures: Algorithms, Design, and Experimental Studies.*

variable shifting(for $2^{-i}$ term), and addition. Notice that to use non-redundant selection function(to know the sign of $Y[i]$), a carry-propagate adder is needed.

For SVD, $\theta_{left}$ and $\theta_{right}$ have to be calculated. For the calculation of these angles, Cavallaro and Luk adopted the direct two angle method; this requires the computation of the intermediate angles $\theta_{sum}$ and $\theta_{diff}$ such that

$$\theta_{sum} = \theta_{left} + \theta_{right} = \tan^{-1}\frac{y+x}{z-w}$$

$$\theta_{diff} = \theta_{right} - \theta_{left} = \tan^{-1}\frac{y-x}{z+w}$$

where $x, y, w, z$ are the elements of the 2 x 2 matrix.

After that, $\theta_{left}$ and $\theta_{right}$ are computed. The computation of these angles is performed using an adder/subtractor and two CORDIC modules as described in Figure 1.

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

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

where, $\sigma_i$ is computed from $Z[i]$

$$\sigma_i = \begin{cases} 1 & \text{if } Z[i] \geq 0 \\ -1 & \text{if } Z[i] < 0 \end{cases} \quad (4)$$

Notice that the recurrence equations of $X$ and $Y$ are the same for both angle and rotation modes, which only require shifters and adders for implementation. To perform the two-sided rotation(with the $\sigma$-encoded values of $\theta_{left}$ and $\theta_{right}$), two CORDIC operations are performed as shown in Figure 1.

**Scaling:** CORDIC changes the length of the vector while rotating. To maintain the same length of the input vector $(X_r[0], Y_r[0])$, the following CORDIC scaling operation is necessary, where $(x^R, y^R)$ is the rotated vector.

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

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

Since two angles (left and right angles) are associated with CORDIC for SVD, the corresponding scale factor becomes $K^2$. Instead of performing the scaling operation by multiplication by $1/K^2$, Cavallaro and Luk proposed a method similar to the ones proposed in [8, 9, 10], which includes additional number of iterations to make the scaling factor $K^2$ equal to 2 and perform the scaling by shifting. For an efficient implementation, the total number of these additional iterations need to be minimized; they showed that there exists a solution with 25% more iterations to force $K^2$ to be 2.

**Evaluation:** The timing diagram is shown in Figure 2. This SVD processor takes $3.25T_C$ cycles where $T_C$ corresponds to the time to complete either an angle calculation or a plane rotation. The three CORDIC operations (one to compute the two angles $\theta_{sum}$ and $\theta_{diff}$ simultaneously, and two to perform the two-sided rotations) take $3T_C$ and the final scaling operation for the two-sided rotation takes $0.25T_C$ cycles.

## 2.2 Redundant CORDIC

**Angle computation and Rotation:** Ercegovac and Lang [11] have shown that the performance of the processor can be improved by using the redundant arithmetic(carry-free addition) and by determining the direction of the rotation based on an estimate instead of an exact value. This requires the modification of the recurrence and selection function to determine the direction of rotation, $\hat{\sigma}$.

In addition, $\hat{\sigma}_i^{left}$ and $\hat{\sigma}_i^{right}$ (corresponding to $\theta_{left}$ and $\theta_{right}$) are obtained on-the-fly using $\hat{\sigma}_i^{sum}$ and $\hat{\sigma}_i^{diff}$, which eliminates $Z$ recurrence since $\theta_{sum}$ and $\theta_{diff}$ do not need to be computed. This computation is done by the following on-line algorithm,

$$W[i] = Z[i] + \hat{\sigma}_{i+p+1} 2^i \tan^{-1}(2^{-(i+p+1)}) \quad (6)$$

$$Z[i+1] = 2(W[i] - \hat{\gamma}_i 2^i \tan^{-1}(2^{-i})) \quad (7)$$

with an initial condition

$$Z[0] = \sum_{j=0}^{p} \hat{\sigma}_j \tan^{-1}(2^{-j}) \quad (8)$$

where $\hat{\sigma}$ belongs to the set $\{ \pm 1, \pm\frac{1}{2}, 0 \}$ and $\hat{\gamma}$ to the set $\{ \pm 1, 0 \}$. The corresponding selection function, which determines $\hat{\gamma}_i$ using an estimate of $W[i]$ and on-line delay $p = 2$ is

$$\hat{\gamma}_i = \begin{cases} 1 & \text{if } \hat{W}[i] \geq \frac{1}{2} \\ 0 & \text{if } -\frac{1}{4} \leq \hat{W}[i] \leq \frac{1}{4} \\ -1 & \text{if } \hat{W}[i] \leq -\frac{1}{2} \end{cases} \quad (9)$$

where $\hat{W}[i]$ is computed using two fractional bits of the redundant representation of $W[i]$.

The computations of these steps are shown in Figure 3. Notice that since $\hat{\gamma}$ is produced from $\hat{\sigma}$, $Z$ recurrence is not needed for CORDIC rotation units, either.

**Scaling:** The same scaling operation as in non-redundant CORDIC has to be performed. However, in redundant CORDIC, the scale factor $K$ is *not a constant* as $\hat{\gamma}_i \in$ {-1, 0, 1}. Therefore, $K$ must be computed for each $\hat{\gamma}$-encoded angle, which has two steps: (1) $K^2$ computation using $P[j+1] = P[j] + |\hat{\gamma}_j| 2^{-2j} P[j]$ with initial condition $P[0] = 1$ and (2) $K = \sqrt{P}$. $K$ computation is overlapped with angle computation to minimize the loss of timing. However, the division operation is needed for the final scaling operation.

**Evaluation:** The timing diagram is shown in Figure 4. This scheme produced a factor of 4 speed-up compared

with the non-redundant scheme [11]. The speed-up is due to the following factors: redundant arithmetic, use of an estimate to determine the direction of the rotation, generation of encoded angles of $\theta_{left}$ and $\theta_{right}$ ($\hat{\gamma}_i$) on-the-fly from encoded angles of $\theta_{sum}$ and $\theta_{diff}$. However, it introduced additional cost since the scale factor is a variable.

# 3 CFR-CORDIC for SVD

In this section, we develop a new redundant scheme, called Constant-Factor-Redundant-CORDIC(CFR-CORDIC) for SVD to reduce the implementation cost of redundant CORDIC.

## 3.1 Development of CFR-CORDIC

**Angle computation:** To compute the direction of the rotations, $\hat{\sigma}_i$(more specifically $\hat{\sigma}_i^{sum}$ and $\hat{\sigma}_i^{diff}$), the modified recurrence equations and selection function for signed-digit representation [11] are used.

$$X[i+1] = X[i] + \hat{\sigma}_i 2^{-2i} W[i]$$
$$W[i+1] = 2(W[i] - \hat{\sigma}_i X[i]) \quad (10)$$
$$\text{where } W[i] = 2^i Y[i] \quad (11)$$

$$\hat{\sigma}_i = \begin{cases} 1 & \text{if } \hat{W}[i] \geq \frac{1}{2} \\ 0 & \text{if } \hat{W}[i] = 0 \\ -1 & \text{if } \hat{W}[i] \leq -\frac{1}{2} \end{cases} \quad (12)$$

where $\hat{W}[i]$ is an estimate computed using one fractional bit of the redundant representation of $W[i]$.

The $\hat{\sigma}$-encoded angles of $\theta_{left}$ and $\theta_{right}$ are obtained again on-the-fly while $\hat{\sigma}$-encoded angles of $\theta_{sum}$ and $\theta_{diff}$ are produced. That is,

$$\hat{\sigma}_i^{left} \tan^{-1} 2^{-i} = \frac{\hat{\sigma}_i^{sum} \tan^{-1} 2^{-i} - \hat{\sigma}_i^{diff} \tan^{-1} 2^{-i}}{2}$$

$$\hat{\sigma}_i^{right} \tan^{-1} 2^{-i} = \frac{\hat{\sigma}_i^{sum} \tan^{-1} 2^{-i} + \hat{\sigma}_i^{diff} \tan^{-1} 2^{-i}}{2}$$

As discussed earlier, the result of direct computation of $\hat{\sigma}_i^{left}$ and $\hat{\sigma}_i^{right}$ from $\hat{\sigma}_i^{sum}$ and $\hat{\sigma}_i^{diff}$ produce the set $\{ \pm 1, \pm\frac{1}{2}, 0 \}$. (Hereafter we will again skip the superscript to simplify the notation.)

To have a constant scale factor, we now need to find a set of $\hat{\gamma}_j$, which is in {-1, 1}, satisfying

$$\sum_{j=0}^{n-1} \hat{\sigma}_j \tan^{-1}(2^{-j}) = \sum_{j=0}^{n-1} \hat{\gamma}_j \tan^{-1}(2^{-j})$$

As in [11] we define the residual $Z$ as:

$$Z[i] = 2^i \left(\sum_{j=0}^{i+p} \hat{\sigma}_j \tan^{-1}(2^{-j}) - \sum_{j=0}^{i-1} \hat{\gamma}_j \tan^{-1}(2^{-j})\right)$$

where p is the on-line delay. The resulting recurrence equation is

$$Z[i+1] = 2(Z[i] + \hat{\sigma}_{i+p+1} 2^i \tan^{-1}(2^{-(i+p+1)}) - \hat{\gamma}_i 2^i \tan^{-1}(2^{-i}))$$

with an initial condition

$$Z[0] = \sum_{j=0}^{p} \hat{\sigma}_j \tan^{-1}(2^{-j}) \quad (13)$$

To simplify the selection function, the recurrence equation is decomposed into two parts by introducing an intermediate variable $W$, as shown before [11],

$$W[i] = Z[i] + \hat{\sigma}_{i+p+1} 2^i \tan^{-1}(2^{-(i+p+1)}) \quad (14)$$
$$Z[i+1] = 2(W[i] - \hat{\gamma}_i 2^i \tan^{-1}(2^{-i})) \quad (15)$$

To have a constant scale factor, the direction of the rotation $\hat{\gamma}$, now need to be in {-1, 1}. However, since $\hat{\gamma}$ is determined from an estimate and from a partial set of $\hat{\sigma}$, the convergence can not be assured.

Takagi et al. [12] proposed *a correcting iteration scheme* to solve the similar problem, which occurs when redundant CORDIC is used to compute cosine and sine function. For these computations, the direction of the rotation $\hat{\sigma}_i$ is obtained from the estimate of $Z[i]$ and is forced to be in {-1, 1} to have a constant scale factor. To assure convergence, some iterations of CORDIC, called *correcting iterations* , are repeated at fixed intervals, where the frequency of repetition depends on the precision of the estimate. Since this scheme is proposed for cosine and sine computation, only the rotation mode of CORDIC is discussed.

We have shown the extension of this scheme to compute angle [13], especially for matrix triangularization [14], which requires to deal with the inter-dependency of the recurrences of $X$ and $Y$.

In the case of CFR-CORDIC for SVD, the problem is aggravated by the fact that not only an estimate of Z[i] is used but its computation is on-line. We use the concept of correcting iterations and on-line delays to solve this problem. In addition, we improve the scheme by minimizing the number of extra correcting iterations by dividing the iterations of CORDIC(for $i = 0$ to $i = n-1$) into two groups: one group where the direction of the rotation is in {-1, 1} for $i = 0$ to $i = \frac{n}{2}$ and the other in {-1, 0, 1} for $i = \frac{n+1}{2}$ to $i = n-1$. With these two groups, we reduce the number of correcting iterations by 50 % since correcting iteration is not needed for the second half of the iterations and we still obtain a constant scale factor $K$ since the value of $K$ in $n$-bit precision does not depend on the value of direction of the rotation for $\frac{n+1}{2} \leq i \leq (n-1)$.

The corresponding selection functions to make the constant scale factor are :

$$(1)\ 0 \leq i < \frac{n}{2}: \quad \hat{\gamma}_i = \begin{cases} 1 & \text{if } \hat{W}[i] \geq 0 \\ -1 & \text{if } \hat{W}[i] < 0 \end{cases} \quad (16)$$

where the estimate $\hat{W}[i]$ is computed using $t$ fractional bits of the redundant representation of $W[i]$. As discussed later, the value of $t$ and on-line delay $p$ shown in Equation 14 depends on the frequency of correcting iterations.

$$(2)\ \frac{n}{2} \le i \le n-1: \quad \hat{\gamma}_i = \begin{cases} 1 & \text{if } \hat{W}[i] \ge \frac{1}{2} \\ 0 & \text{if } -\frac{1}{4} \le \hat{W}[i] \le \frac{1}{4} \\ -1 & \text{if } \hat{W}[i] \le -\frac{1}{2} \end{cases} \tag{17}$$

Notice that the selection function of the second half iterations is the same one as shown in Equation 9.

This selection function forces the scale factors for the two-sided rotations to be constant but does not satisfy the convergence requirement. In fact, the convergence requirement was satisfied in the previous redundant scheme by setting the $\hat{\gamma}$ to be 0 for the region $U_{-1} < W[i] < L_1$ (more accurately, by forcing $L_0 \le U_{-1}$ and $L_1 \le U_0$) where $L_1 = 2^i \tan^{-1}(2^{-(i+p+1)})$ and $U_{-1} = -L_1$. This interval $[L_k, U_k]$ of $W[i]$ for the redundant scheme of Ercegovac and Lang was determined such that $Z[i+1]$ remained bounded when choosing $\hat{\gamma}_i = k$.

As we already pointed out, $\hat{\gamma}$ must not be zero to have a constant scale factor. Furthermore, the region to select 1 for $\hat{\gamma}$ is bigger than that to select -1 due to the use of the estimate. More specifically, for the region $-2^{-t} < W[i] \le 0$, $\hat{\gamma}$ is selected to be 1. The worst case(where the amount outside of the bound for $W[i+1]$ is maximum) occurs when $W[i] = -2^{-t} + \epsilon$. The resulting $Z[i+1]$ and $W[i+1]$ are

$$Z[i+1] = -2 - 2 * 2^{-t} \quad \text{since } \hat{\gamma}_i = 1$$

$$W[i+1] = -2 - 2 * 2^{-t} - 2^{-(p+1)}$$

assuming $\sigma_{i+p+2} = -1$ for the worst case

In the following iterations, the amount outside of the bound increases and $W[i+m]$ becomes:

$$\begin{aligned} W[i+m] &= -2 - 2^m 2^{-t} - 2^{m-1}2^{-(p+1)} \\ &\quad -2^{m-2}2^{-(p+1)} - \ldots - 2^{-(p+1)} \end{aligned}$$

To recover the bound, we use here a correcting iteration defined as:

$$Z[i+m]^c = W[i+m] - \hat{\gamma}^c_{i+m} 2^{i+m+1} \tan^{-1} 2^{-(i+m)}$$

$$W[i+m]^c = Z[i+m]^c$$

where $\hat{\gamma}^c_{i+m}$ is determined by the same selection function shown in Equation 16. After this, $Z[i+m+1]$ is obtained by the original equation 15 using $W[i+m]^c$ instead of $W[i+m]$. Now to satisfy the convergence requirement, $W[i+m]^c \ge L_{-1}$ must be forced.[1] From this condition, we get

$$2^{-t} + 2^{-(p+1)} < 2^{1-m} \tag{18}$$

[1] $L_{-1} = -2*2^{i+m}\tan^{-1}2^{-(i+m)} + 2^{i+m}\tan^{-1}(2^{-(i+m+p+1)})$ ,as determined in [11].

By solving this, we can find out the relationship among $t$(the number of fractional bits), $p$(the online-delay), and $m$(the frequency indicator for a correcting iteration). We would like to maximize the value $m$ so as not to perform a correcting iteration too often. At the same time, we want to minimize the value $t$ for an efficient implementation of the selection function. A solution of $m = p+1$ and $t = p+2$ satisfies these restrictions. For an example, with on-line delay $p = 3$, a correcting iteration must occur with an interval of 4 or less and 5 fractional bits are used for the estimate. The computation steps are shown in Figure 5.

**Rotation:** As shown in Figure 5, since $\hat{\gamma}$ is produced from $\hat{\sigma}$, $Z$ recurrence is not needed again for CORDIC rotation units.

The rotation processor can also incorporate two schemes we developed to speed up its processing. The first one is to reduce the number of iterations in CORDIC rotation unit by about 25% by expressing the direction of the rotation in radix-4 $\hat{\gamma}$ for the second half of the iterations: this method is based on the fact that for $i \ge \frac{n}{2}$, $\tan^{-1} 2^{-i} = 2^{-i}$ in $n$-bit precision. The second one is to convert the redundant number representation of the rotated output into the conventional number representation on-the-fly, not using a carry-propagate adder. The detailed development of these two schemes can be found in [13]. The incorporation of these schemes into an application system, specifically for matrix triangularization, was discussed in [14].

**Scaling:** When the final scaling operation is needed, it can be simply a shifting by forcing the scale factor $K$ to be 2 (similar to the work for non-redundant CORDIC mentioned earlier), which requires to repeat iterations of CORDIC and scaling iteration of the following form:

$$X[i] = X[i] + \beta_j X[i] 2^{-j}$$

$$Y[i] = Y[i] + \beta_j Y[i] 2^{-j}\ ,\ \beta_j \in \{-1, 1\}$$

The scaling iteration manipulates the extension or reduction of the vector norm without changing $\tan^{-1}\frac{X[i]}{Y[i]}$. Consequently, the scaling iteration does not affect the output of CORDIC angle unit. Moreover, when the precision of the output is fixed, $\beta_j$ can be pre-computed.

For an efficient implementation, we need to minimize the number of additional iterations of CORDIC and scaling iterations. As discussed in [13], the problem of searching for the minimum number of additional iterations for CFR-CORDIC is more complicated(time-consuming) than the one in non-redundant CORDIC since the CFR-CORDIC requires *correcting iterations* to assure convergence. We developed an efficient searching method, called decomposed search, to reduce the searching time from $T$ to $\sqrt{T}$ [13].

## 3.2 Performance analysis

**Evaluation:** Consider the case of $n = 16$ (16-bit precision data) and 5 fractional bits in the estimate. From the condition 18, we know that the interval of a correcting iteration is 4 or less, $m = 4$. Under these conditions,

| Type | Total-time | $n = 16$ |
|---|---|---|
| Non-redundant | $16.25n$ | 260 |
| Redundant CORDIC | $5n + 10$ | 90 |
| CFR-CORDIC | $4n + \alpha$ | 84 |

Table 1: Time comparisons (basic cycles)

| Type | Total-time | $n = 16$ |
|---|---|---|
| Non-redundant | $13.75n$ | 220 |
| Redundant CORDIC | $4n + 10$ | 74 |

Table 2: Time with radix-4 scheme(basic cycles)

| Type | Area estimate |
|---|---|
| Non-redundant CORDIC | 3 |
| Redundant CORDIC | 6.4 |
| CFR-CORDIC | 3.4 |

Table 3: Area comparisons (units)

the minimum number of additional iterations is searched to force the scale factor to be 2 using *decomposed search* , which results in 6 additional iterations: repetitions of the 4th, the 8th and the 9th CORDIC iterations and scaling iterations of {+2, +10, -5 }.

The corresponding overall system timing is shown in Figure 6. As can be seen in the diagram, for $n = 16$ and $m = 4$, the timing of both processors matches well, resulting in an overall time of 42 iterations of CORDIC.

**Comparisons:** To compare the various schemes we need to define a basic cycle. As in [11], we use as this basic cycle the time taken by an iteration similar to radix-2 SRT division, that is, consisting of a 3-1 multiplexer, a carry-save adder, a simple selection function, and the loading of a register. Using this basic cycle, we determine in each scheme the number of basic cycles to complete one iteration of CORDIC and the total time to complete two-sided plane rotations for SVD, namely, angle computations($\theta_{left}$ and $\theta_{right}$) and two-sided plane rotations.

For non-redundant scheme, we estimate that one iteration takes 5 basic cycles since variable shifting and carry-propagate addition need to be performed. Consequently, $T_C$ takes $5n$ basic cycles and the total time takes $16.25n$ basic cycles. For redundant CORDIC with variable scaling, one iteration takes 2 basic cycles as discussed in [11], resulting in an overall time of $5n + 10$ basic cycles shown in the timing diagram of Figure 4. For CFR-CORDIC, assuming that the number of fractional bits for the estimate is chosen not to increase the step time (that is, the selection will be done within the time of variable shifting), one iteration again takes 2 basic cycles. In this case, the total number of iterations depends on $n$(precision of data) and $m$(frequency of correcting iterations). Consequently, the overall time becomes $4n + \alpha$, where $4n$ corresponds to the two-sided rotations and $\alpha$ to the correcting iterations and online delays. For 16-bit precision and an estimate of 5 fractional bits, the CFR-CORDIC scheme takes 42 iterations, resulting in 84 basic cycles. Table 1 summarizes the time comparisons, which shows that the CFR-CORDIC scheme is the fastest scheme for $n = 16$.

The performance of non-redundant and redundant CORDIC can also be improved by applying our scheme with radix-4 direction of the rotation(for the second half of the iterations) since it reduces the total number of iterations as shown in Table 2. In this case, the scheme with variable scaling is faster.

To estimate the area requirement in each scheme, let's define the basic unit to be the area consumed by one conventional CORDIC unit to perform one iteration of angle computation with $X$, $Y$, and $Z$ recurrence equations. We also assume that the CORDIC rotation unit used for the left angle is reused for the right angle.

Then, the non-redundant scheme takes about 3 units since it requires 3 CORDIC units. For redundant CORDIC, all 3 CORDIC units do not have $Z$ recurrence equations and carry-free additions are employed, resulting in 2.6 units for CORDIC units. However, additional areas are necessary to compute $\hat{\gamma}$ on-the-fly(0.8 unit used twice for left and right in sequence) and to compute $K_{left}$(1 unit) and $K_{right}$(1 unit), on-line computation of $K_{left} * K_{right}$ and division for final scaling (1 unit). The resulting area consumption for redundant CORDIC becomes 6.4 units. In the case of CFR-CORDIC, we again require 2.6 units for CORDIC modules similar to the redundant case. Additional area is now for $\hat{\gamma}$ computations, which is estimated to be 0.8. Then, the total area consumption becomes 3.4. These area estimates are summarized in Table 3.

## 4 Conclusion

We briefly reviewed the non-redundant and redundant CORDIC schemes for SVD and presented the Constant-Factor-Redundant-CORDIC(CFR-CORDIC) scheme where the scale factor is forced to be constant while computing angles for plane rotations.

It is important to have a constant scale factor for the SVD CORDIC processor since multiplying a constant to the matrix does not change the output of CORDIC angle unit. In fact, when we skip the final scaling operation for all the plane rotations, the final singular values are multiplied by a constant term $K^r$, where $K$ is the constant factor of a plane rotation and $r$ is total number of plane rotations for SVD. Consequently, with the floating-point implementation, we can completely eliminate the final scaling operation for SVD applications where only the ratio of singular values need to be found. When we need to find singular values themselves, we can skip the final scaling operation for each plane rotation and perform once with $K^r$ at the end. Notice that

with the fixed-point implementation, the scaling operation is required in each two-sided rotation to avoid an overflow/precision error.

In addition, we presented schemes to make the scaling factor equal to 2 by including additional scaling iterations and CORDIC iterations, to reduce the number of iterations in the rotation by using radix-4 direction of the rotation, and to convert on-the-fly from the redundant output to a conventional implementation. We compared this scheme with previously proposed ones and showed that it provides similar execution time as redundant CORDIC with variable scaling factor with significant saving in area.

## References


[1] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1983.

[2] K. Bromley and J.M. Speiser. Signal Processing Algorithms, Architectures, and Applications. *Tutorial 31, SPIE 27th Annual Internat. Tech. Symp.*, 1983.

[3] R.P. Brent, F.T. Luk, and C.F. Van Loan. Computation of the Singular Value Decomposition using Mesh-Connected Processors. *Journal of VLSI and Computer Systems*, 1(3):242–270, 1985.

[4] F.T. Luk. Architectures for Computing Eigenvalues and SVDs. *SPIE Highly Parallel Signal Processing Architectures*, 614:24–33, 1986.

[5] J.R. Cavallaro and F.T. Luk. CORDIC Arithmetic for an SVD Processor. *Proceeding of 8th Symposium on Computer Arithmetic*, :113–120, 1987.

[6] J.E. Volder. The CORDIC Trigonometric Computing Technique. *IRE Trans. Electronic Computers*, EC-8:330–334, September 1959.

[7] J.S. Walther. A Unified Algorithm for Elementary Functions. *AFIPS Spring Joint Computer Conference*, :379–385, 1971.

[8] G. L. Haviland and A. A. Tuszynski. A CORDIC Arithmetic Processor Chip. *IEEE Transactions on Computers*, C-29(2):68–79, February 1980.

[9] H.M. Ahmed. *Signal Processing Algorithms and Architectures*. Ph.D. Dissertation, Department of Electrical Engineering, Stanford University, 1982.

[10] J.M. Delosme. VLSI Implementation of Rotations in Pseudo-Euclidean Spaces. In *Proceedings of IEEE Int. Conf. Acoustics, Speech, and Signal Processing 2*, pages 927–930, 1983.

[11] M. Ercegovac and T. Lang. Redundant and On-line CORDIC: Application to Matrix Triangularization and SVD. *IEEE Transactions on Computers*, C-39(6):725–740, June 1990.

[12] N. Takagi, T. Asada, and S. Yajima. *Redundant CORDIC methods with a Constant Scale Factor for Sine and Cosine Computation*. Submitted to IEEE Trans. on Computers, 1989.

[13] J. Lee. *Redundant CORDIC:Theory and its Application to Matrix Computations*. Ph.D. Dissertation, Computer Science Department, University of California, 1990.

[14] J. Lee and T. Lang. Matrix Triangularization by Fixed-Point Redundant CORDIC with a Constant Scale Factor. *Proc. SPIE Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations*, July 1990.


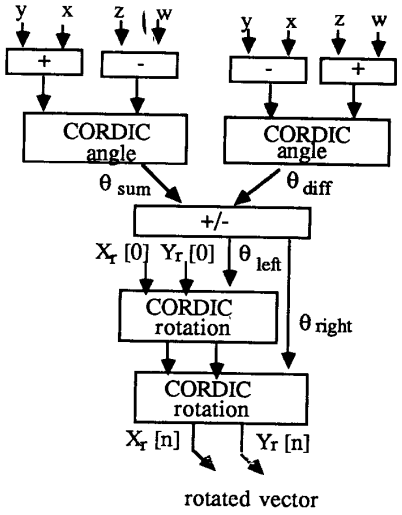


Figure 1: Non-redundant CORDIC scheme

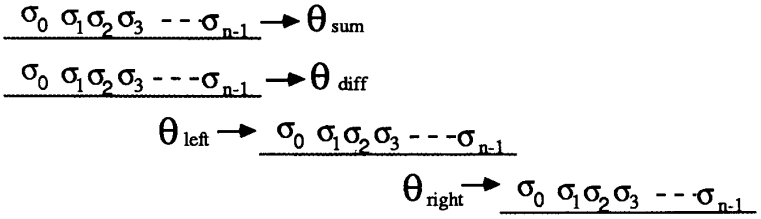


Figure 2: Timing of non-redundant scheme

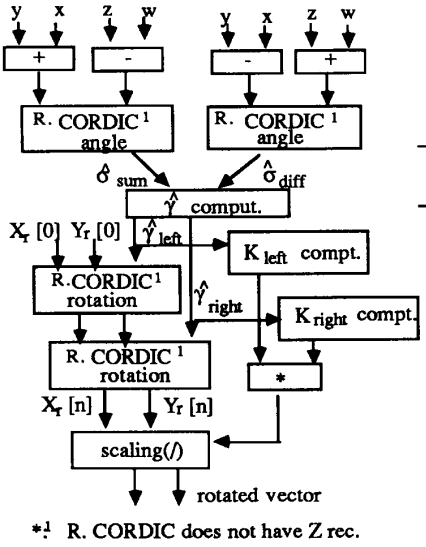


Figure 3: Redundant CORDIC(var. scale factor)

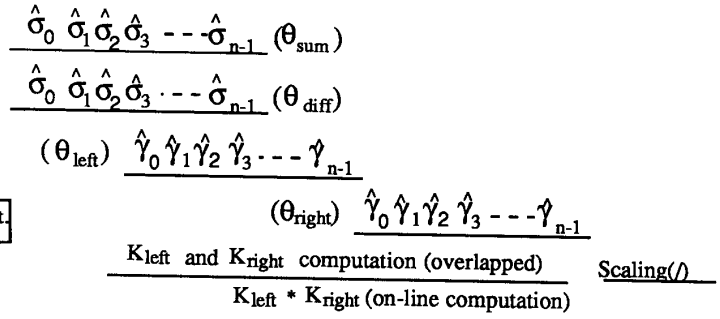


Figure 4: Timing of redundant CORDIC

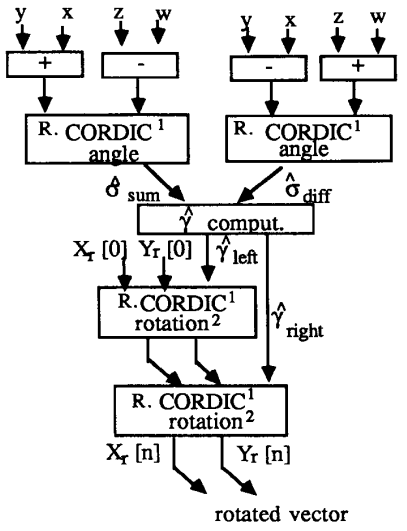


Figure 5: CFR-CORDIC

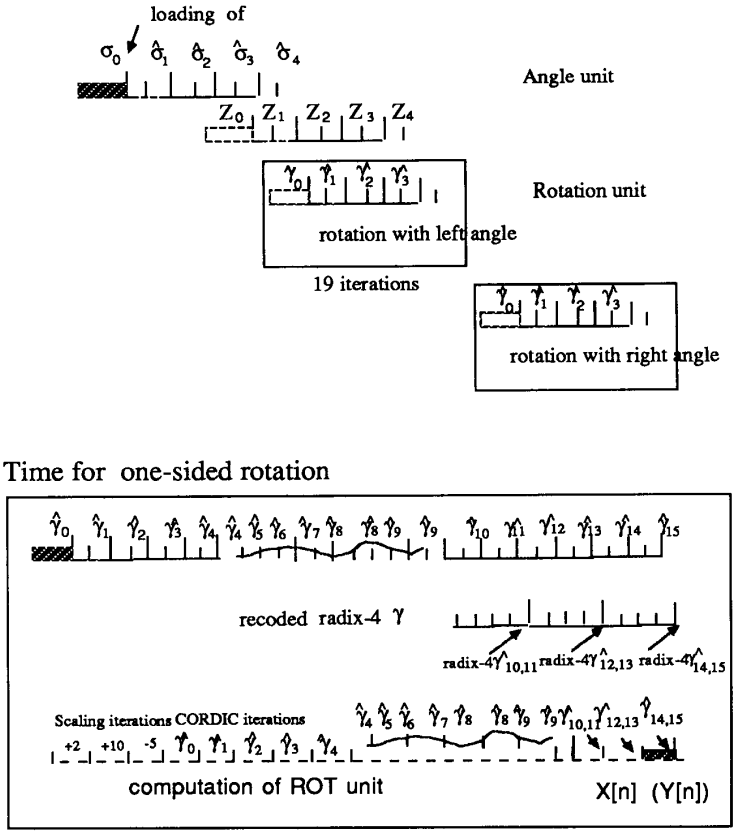


Figure 6: CFR-CORDIC timing for SVD