Journal of VLSI Signal Processing 33, 19–29, 2003

Fast CORDIC Algorithm Based on a New Recoding Scheme for Rotation Angles and Variable Scale Factors

JEN-CHUAN CHIH AND SAU-GEE CHEN
Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan, ROC

Received November 2, 2000; Revised July 17, 2001

Abstract. This work proposes a new rotation mode CORDIC algorithm, which considerably reduces the iteration number. It is achieved by combining several design techniques. Particularly, a new table-lookup recoding scheme for rotation angles and variable scale factors is developed to reduce the iteration numbers for rotation and scale factor compensation. By addressing the MSB parts of the residual rotation angles to a lookup table, two micro rotation angles are retrieved that in combination best matches the MSB parts. We also combine the leading-one bit detection operations for residual rotation angles, to skip unnecessary rotations. The resulting problems of variable scale factors are then solved by our previous fast decomposition and compensation algorithm (C.C. Li and S.G. Chen, in *Proceedings of 1996 IEEE International Symposium Circuits and Systems*, May 1996, Atlanta, USA, pp. 264–267; C.C. Li and S.G. Chen, in *Proceedings of 1997 IEEE International Conference on Acoustic, Speech and Signal Processing*, Munich, 1997, Germany, pp. 639–642). To further reduce the iteration number of scale factor compensation, we again apply the mentioned residual recoding technique and the leading-one bit detection scheme to the fast variable scale factor algorithm. Those techniques collectively reduce the iteration number significantly. Simulations show that in average the new design needs only 9.78 iterations to generate results with 22-bit accuracy, including all the iterations for rotations and scale factor compensations. Statistically, the total iteration number is less than $n/2$ for results with n-bit accuracy. The introduced extra table size is of the same order of magnitude as that for the angle set $\{\tan^{-1} 2^{-i}, i = 0, 1, \ldots, n\}$, required by general CORDIC algorithms. The new recoding scheme can be applied to other elementary function such as division and square-root functions.

Keywords: CORDIC algorithm, variable scale factor, table lookup method, residual angle and scale factor recoding

1. Introduction

The basic CORDIC (Coordinate Rotation Digital Computer) algorithm [1, 2] is a highly efficient iterative technique for vector rotations, de-rotations, as well as for the computation of elementary functions. Since the algorithm can be realized as a sequence of shift-and-add operations, it is very suited for VLSI implementation and popular for general engineering and DSP applications.

There have been numerous improved CORDIC algorithms and structures proposed ever since its introduction. Most of the CORDIC algorithms assume a constant scale factor for the ease of scale factor compensation. However, they have to rotate even when the residual rotation angles have converged [3–6]. In some cases, they either have to do accurate but slow decision operations for rotation directions or do rough direction decisions at the expense of extra compensation operations [3, 4]. To speedup CORDIC operations, the following techniques are widely used: (1) use carry-free redundant addition scheme [3, 4, 7–10]; (2) fast decision of rotation directions with only a few most significant digits (MSDs) of the control parameters [3, 4, 7–10]; (3) skip unnecessary rotations; (4) effectively recode rotation angles for saving rotation iterations [11]; (5) apply radix-4 rotation schemes [8, 12–14], to reduce iteration numbers; and (6) predict the rotation sequence for parallel and pipelined processing [15–18].

Some of the mentioned techniques result in variable scale factors. Variable scale factors have the trouble of complicated scale factor computation followed by penalty compensation [9, 10]. Due to the considerable overhead generated by variable scale factor, most of the existing radix-4 CORDIC algorithms resort to constant scale factor approach [8, 13]. However, these constant-scale-factor CORDICs are basically hybrid radix-2 and radix-4 algorithms. As a result, their iteration numbers are not fully reduced. Recently, we proposed CORDIC algorithms with variable scale factors [12, 16, 19], skip unnecessary rotations and at the same time perform low-complexity on-line decompositions and compensations for the variable scale factors. Specifically, the radix-4 algorithm costs fewer iterations (including rotations and compensations) than the existing radix-4 algorithms. The radix-4 CORDIC algorithm proposed in [14] is similar to the one in [12], except the ways they handle variable scale factors. Both designs share the same low iteration number of $0.8n$. Note that a similar variable scale factor decomposition and compensation algorithm to that in [12, 19] was proposed much later in [20, 21]. Although the very high-radix CORDIC algorithm [20, 21] has an extremely small iteration number, it is irregular in realization, which needs multiplication-and-accumulation circuits. Its efficiency is high dependent on practical circuit optimization.

In this paper, to further reduce the shift-and-add operations of both rotation iterations and scale factor compensations, we will present a new table lookup recoding scheme for rotation angles and variable scale factors. The new method can speedup both the convergence rates of the residual rotation angles and our fast variable scale factor decomposition and compensation algorithm [12, 19]. For more reduction of iteration number, the new CORDIC algorithm also applies the leading-one bit detection operations to both residual rotation angles and decomposition of variable scale factors.

This paper is organized as follows. Section 2 reviews the basic CORDIC algorithm, while Section 3 proposes a new recoding algorithm for residual rotation angles and exponents, followed by the introduction of new CORDIC algorithm. Evaluation and simulation of the new CORDIC algorithm are conducted in Section 4. Section 5 draws a brief conclusion.

2. Basic CORDIC Algorithm

CORDIC algorithm is a generalized algorithm that can perform vectoring and rotation operations of a vector in three different kinds of coordinate systems, including (1) the linear coordinate system, (2) the circular coordinate system, and (3) the hyperbolic coordinate system. The operations are approached by a sequence of micro-rotations using only shift-and-add operations, combined with table-lookup operations. In the paper, we only focus on the most popular circular coordinate system. Similar results can be directly applied to the other two systems. In addition, we only investigate the rotation mode operations. Algorithm for vectoring mode can be similarly derived. First, the basic theory of the CORDIC algorithm is reviewed as follows.

Given a vector $p(x, y)$, to be rotated by an angle θ, the rotated output vector $p(x', y')$ is

$$x' = x\cos\theta + y\sin\theta \tag{2.1}$$

$$y' = y\cos\theta - x\sin\theta \tag{2.2}$$

One can take the out the cosine factor and approximate θ by $\theta \approx \sum_{i=0}^{n} \delta_i\theta_i$, where $\delta_i = \{-1, 1\}$ and $\theta_i = \tan^{-1} 2^{-i}$. Therefore, in matrix form

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \approx K \prod_{i=0}^{n} \begin{bmatrix} 1 & \delta_i 2^{-i} \\ -\delta_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.3}$$

where

$$K = \prod_{i=0}^{n} \cos\theta_i = \prod_{i=0}^{n} \frac{1}{\sqrt{1+2^{-2i}}} = \prod_{i=0}^{n} K_i \tag{2.4}$$

Decision of $\delta_i \in \{-1, 1\}$ is for the convergence of the residual angle z_{i+1} to zero:

$$z_{i+1} \equiv z_i - \delta_i\theta_i \equiv \theta - \sum_{m=0}^{i} \delta_m\theta_m \quad \text{and} \quad \delta_i = \operatorname{sign}(z_i) \tag{2.5}$$

As such, computation of final x' and y' consists of two parts. One is the iteration of the shift-and-add operations,

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & \delta_i 2^{-i} \\ -\delta_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \left(\prod_{m=1}^{i} \begin{bmatrix} 1 & \delta_m 2^{-m} \\ -\delta_m 2^{-m} & 1 \end{bmatrix} \right) \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \quad i = 0, 1, \ldots, n \tag{2.6}$$

followed by the second part of scale factor compensation:

$$x'_{n+1} = K \cdot x_{n+1}, \quad y'_{n+1} = K \cdot y_{n+1} \tag{2.7}$$

Here $(x_0, y_0) = (x, y)$ is the initial vector, and (x'_{n+1}, y'_{n+1}) is the final result after $n + 1$ rotations.

In total, the algorithm needs $n + 1$ and $n/3$ to $n/4$ shift-and-add operations for the rotations and scale factor compensation, respectively. The number of rotation can be further reduced by skipping all the unnecessary rotations of $\theta_i = \tan^{-1} 2^{-i}$ up to $\theta_{m-1} = \tan^{-1} 2^{-(m-1)}$, if the leading-one bit (or leading-zero bit) of the positive (or negative) residual z_i is the m-th most significant fraction bit. In that case we can set $\delta_i = \delta_{i+1} = \cdots = \delta_{m-1} = 0$. The reduction in rotation number introduces the nasty problem of variable scale factors. The cost normally exceeds the saving in rotation number. It is the main disadvantage of the existing CORDIC algorithms with variable scale factors. To alleviate the problem, we developed a simple and efficient algorithm [12, 19] for variable scale factor decomposition and compensation. The algorithm will be further improved and included in the new CORDIC algorithm as will detailed next.

3. Fast CORDIC Algorithm Based on a New Table-Based Residual Recoding Method

For fast convergence, first we detect the leading-one (leading-zero) bit positions, for positive (negative) residual angle z_i, respectively, in the i-th iteration. This action avoids unnecessary rotations required by conventional CORDIC algorithms. Then the most significant r bits (denoted as $z_{i,r}$), counted from the leading-one (or leading-zero) bit of z_i, are used to access δ_{m_i} and δ_{l_i} information from a table, where $\delta_{m_i}, \delta_{l_i} \in \{1, -1\}$ and $m > l$. These two retrieved parameters correspond to a combined rotation angle $\delta_{m_i} \tan^{-1} 2^{-m_i} + \delta_{l_i} \tan^{-1} 2^{-l_i}$ that best matches $z_{i,r}$ (in a least-square error sense), which makes $z_{i,r} - (\delta_{m_i} \tan^{-1} 2^{-m_i} + \delta_{l_i} \tan^{-1} 2^{-l_i})$ as close to zero as possible. This approach corresponds to the following iteration operation.

$$\begin{aligned} x_{i+1} &= x_i \left(1 - \delta_{l_i} \delta_{m_i} 2^{-(l_i + m_i)}\right) + y_i \left(\delta_{l_i} 2^{-l_i} + \delta_{m_i} 2^{-m_i}\right) \\ y_{i+1} &= y_i \left(1 - \delta_{l_i} \delta_{m_i} 2^{-(l_i + m_i)}\right) - x_i \left(\delta_{l_i} 2^{-l_i} + \delta_{m_i} 2^{-m_i}\right) \end{aligned} \tag{3.1}$$

This iteration results in a variable scale factor.

$$K = \prod_{i=1}^{I} \cos \theta_{l_i} \cos \theta_{m_i} = \prod_{i=1}^{I} \frac{1}{\sqrt{1 + 2^{-2l_i}}} \frac{1}{\sqrt{1 + 2^{-2m_i}}} \tag{3.2}$$

The four-operand addition can be easily reduced to conventional two-operand addition by using carry-save adders. The best combination of δ_{m_i} and δ_{l_i} is done by computer search, which is dependent on $z_{i,r}$, and the range size q of δ_{m_i} and δ_{l_i}.

In generalization, we may include more than two δ_k's and a larger q to speedup the convergence rate. However, the computational complexity increases significantly. Hence, here we only investigate the case of two combined direction parameters. Similar techniques can be extended to the general case. The variable scale factor described by (3.2) introduces considerable overhead in the existing CORDIC algorithms. To solve the problem, we will combine our previously proposed efficient decomposition and compensation algorithm for variable scale factors [12, 19]. In order to further reduce the iteration number, we will apply the same residual recoding technique to this scale factor algorithm. In the end, a fast CORDIC algorithm with a small combined iteration number can be obtained, as will be detailed next.

3.1. *Recoding Tables for the Residual Rotation Angles*

Based on (3.1), some lookup tables for the residual rotation angles can be constructed by computer search with the closest match as mentioned before. In a sense, it approximately amounts to a radix-2^r CORDIC algorithm, by examining the MSB part $z_{i,r}$ of the residual rotation angle z_i. Since an optimal table depends on the iteration index i, it is better to have an optimized lookup table for each i. However, it will increase the table size accordingly. From computer simulations, we find that it is enough to have good results by using only two different tables, as shown below.

Table 1. Recoding table for the residual rotation angle, Case (1): $k = \{0, 1\}, r = 4, q_0 = 8$.

$z_{i,r}$	Optimized patterns of δ_{l_i} and δ_{m_i}							
$(2^{-k} \sim 2^{-(k+3)})$	δ_{k-1}	δ_k	δ_{k+1}	δ_{k+2}	δ_{k+3}	δ_{k+4}	δ_{k+5}	δ_{k+q-2}
1000	0	0	1	0	1	0	0	0
1001	0	1	0	−1	0	0	0	0
1010	0	0	1	1	0	0	0	0
1011	0	1	0	0	−1	0	0	0
1100	0	1	0	0	0	−1	0	0
1101	0	1	0	0	0	0	0	1
1110	0	1	0	0	0	1	0	0
1111	1	0	−1	0	0	0	0	0

Table 2. Recoding table for the residual rotation angle, Case (2): $k > 1, r = 4, q_1 = 7$.

$z_{i,r}$	Optimized patterns of δ_{l_i} and δ_{m_i}						
$(2^{-k} \sim 2^{-(k+3)})$	δ_k	δ_{k+1}	δ_{k+2}	δ_{k+3}	δ_{k+4}	δ_{k+5}	δ_{k+q-1}
1000	0	1	0	0	1	0	0
1001	0	1	0	1	0	0	0
1010	0	1	1	0	0	0	0
1011	0	1	1	0	0	0	0
1100	1	0	0	−1	0	0	0
1101	1	0	0	0	−1	0	0
1110	1	0	0	0	0	0	1
1111	1	0	0	0	1	0	0

Here, we take $r = 4$ bits (i.e., radix-2^4) as a design example. Table 1 shows the stored optimized δ_{m_i} and δ_{l_i} patterns, corresponding to the $z_{i,r}$ information. From Taylor's expansion of $\theta_k = \tan^{-1} 2^{-k}$, we find that the binary patterns of θ_0 and θ_1 are noticeably different from those of the θ_k's, $k > 1$. Therefore, two different tables are used for the cases of $k = \{0, 1\}$ and $k > 1$, respectively, where k is the leading-one (or leading-zero) bit position of the residual rotation angle z_i. Simulations show that this kind of arrangement produces good results close to the optimized solution. This configuration has a table size (in bits) of $2(q_0 + q_1) \times 2^{r-1} = (q_0 + q_1) \times 2^r$, where q_0 and q_1 are the δ_i range sizes corresponding to the cases of $k = \{0, 1\}$ and $k > 1$, respectively, and 2^r is the radix number.

Note that the given table size does not include the term $n \times p$ for $\{\tan^{-1} 2^{-i}, i = 0, 1, \dots, n\}$, required by conventional CORDIC algorithms. The extra table size $(q_0 + q_1) \times 2^r$ is small, because r and q are generally much less than n. Note that when roughly $k > n/3$, δ_{m_i} and δ_{l_i} patterns can be fully decided by the residual rotation angles, such that the table size can be further reduced. Also note that we can have a faster convergence rate, at the cost of a larger range number and table size.

Example. Consider the radix-16 (2^4) residual angle recoding algorithm as shown in Tables 1 and 2. In the tables, each row has q entries indicating the presence ($\delta_i = \pm 1$) or absence ($\delta_i = 0$) of δ_i. Here, at most we only allow two none-zero δ_i's, which corresponds to δ_{l_i} and δ_{m_i} of (3.1). For better residue angle convergence, we separate the recoding scheme into two cases, depending on the leading-one bit position k of the residue angle as shown in Tables 1 and 2.

3.2. *Fast Variable Scale Factor Decomposition Algorithm Based on the Residual Recoding Scheme*

For low-complexity decomposition and compensation of the variable scale factors described by (3.2), here we further improve and speedup our previous efficient variable scale factor algorithm [12, 19], by using a residual recoding scheme similar to that for residual rotation angle. The whole improved algorithm is detailed below.

According to [12, 19], K can be first transformed to

$$\begin{aligned} T \equiv \ln K &= \prod_{i=1}^{I} \cos\theta_{l_i} \cos\theta_{m_i} \\ &= \ln\left[\prod_{i=1}^{I} \frac{1}{\sqrt{1+2^{-2l_i}}} \frac{1}{\sqrt{1+2^{-2m_i}}}\right] \\ &= -\frac{1}{2}\sum_{i=1}^{I}\left[\ln\left(1+2^{-2l_i}\right) + \ln\left(1+2^{-2m_i}\right)\right] \quad (3.3) \end{aligned}$$

This equation suggests that from the information of δ_{m_i} and δ_{l_i}, we can obtain the terms $-\frac{1}{2}\ln(1 + 2^{-2l_i})$ and $-\frac{1}{2}\ln(1 + 2^{-2m_i})$ from a lookup table and perform the accumulation operation of (3.3), in parallel with the rotation iterations. In the end of all the rotation operations, T is also solved. The scale factor K is obviously equal to $e^T \cdot e^T$ in turns can be decomposed into a sequence of shift-and-add terms, by using the CCM algorithm [22].

To speedup the decomposition operation, we apply the same residual recoding technique to reduce the exponent residual successively. Specifically, we first

Table 3. Recoding table for the residual exponents of scale factors.

$\beta_{j,s}$ $(2^{-k} \sim 2^{-(k+3)})$	Optimized patterns of ρ_{l_j} and ρ_{m_j}				
	ρ_{k-1}	ρ_k	ρ_{k+1}	ρ_{k+2}	ρ_{k+q-2}
1000	0	1	0	0	0
1001	0	1	0	0	1
1010	0	1	0	1	0
1011	0	1	1	0	0
1100	0	1	1	0	0
1101	1	0	0	−1	0
1110	1	0	0	−1	0
1111	1	0	0	0	−1

perform the leading-one (or zero) bit detection of the residual exponent β_j in the j-th iteration. Then the s MSB's $\beta_{j,s}$ (counted from the leading bit) of β_j are used as the address to retrieve the terms $\ln(1+\rho_{l_j}2^{-l_j})$ and $\ln(1+\rho_{m_j}2^{-m_j})$ from a table. The table is optimized such that $\beta_{i+1} = \beta_i - \ln(1+\rho_{l_j}2^{-l_j}) - \ln(1+\rho_{m_j}2^{-m_j})$ is as close to zero (in a least-square error sense) as possible. As such, the corresponding decomposition equation is:

$$e^T = K \approx \prod_{j=1}^{J}\left(1+\rho_{l_j}2^{-l_j}\right)\left(1+\rho_{m_j}2^{-m_j}\right) \quad (3.4)$$

where $\rho_{l_j}, \rho_{m_j} \in \{-1, 1\}$. Table 3 lists the optimized patterns of ρ_{l_j} and ρ_{m_j}, for the example of $s = 4$, $q = 5$.

Once the optimized ρ_{l_j} and ρ_{m_j} combination is retrieved, they are used to access the corresponding terms $\ln(1+\rho_{l_j}2^{-l_j})$ and $\ln(1+\rho_{m_j}2^{-m_j})$ from another lookup table. These two terms are then subtracted from the exponent residual. Similarly, to speedup the convergence rate of the decomposition, we can have more than two non-zero ρ_i's and a larger q in the lookup table, at the cost of higher complexity per iteration. The required table size is similar to that for residual rotation angles.

3.3. *The New CORDIC Algorithm*

In summary, by combing the leading-bit detection scheme, the residual recoding technique, and the fast decomposition and compensation algorithm for variable scale factors, we have a CORDIC algorithm as detailed by the following steps:

(1) Set the initial iteration number $i = 0$, initial residual angle $z_0 = \theta$, initial rotation vector $(x_0, y_0) = (x, y)$, and initial exponent residual $T_0 = 0$. If $\theta = 0$, then $(x', y') = (x, y)$; end of the rotation and exit the iteration. Otherwise, proceed to step (2).

(2) Check leading-one bit position k and obtain $z_{i,r}$ of z_i. If $z_i \neq 0$, go to step (3). Otherwise $z_i = 0$: rotation operations are completed; set the total iteration number $I = i - 1$; go to step (5), for further decompositions and compensations of variable scale factors.

(3) Using $z_{i,r}$, retrieve the optimized δ_{l_i} and δ_{m_i} information, $-\frac{1}{2}\ln(1+2^{-2l_i})$, $-\frac{1}{2}\ln(1+2^{-2m_i})$, $\tan^{-1}2^{-l_i}$, and $\tan^{-1}2^{-m_i}$ from lookup tables, and perform the iteration:

$$\begin{aligned} x_{i+1} &= x_i\left(1-\delta_{l_i}\delta_{m_i}2^{-(l_i+m_i)}\right) \\ &\quad + y_i\left(\delta_{l_i}2^{-l_i}+\delta_{m_i}2^{-m_i}\right) \\ y_{i+1} &= y_i\left(1-\delta_{l_i}\delta_{m_i}2^{-(l_i+m_i)}\right) \\ &\quad - x_i\left(\delta_{l_i}2^{-l_i}+\delta_{m_i}2^{-m_i}\right) \end{aligned} \quad (3.5)$$

$$z_{i+1} = z_i - \left(\delta_{m_i}\tan^{-1}2^{-m_i} + \delta_{l_i}\tan^{-1}2^{-l_i}\right) \quad (3.6)$$

$$\begin{aligned} T_{i+1} = T_i + \Bigg[&\left(-\frac{1}{2}\right)\ln(1+2^{-2l_i}) \\ &+\left(-\frac{1}{2}\right)\ln(1+2^{-2m_i})\Bigg] \end{aligned} \quad (3.7)$$

(4) Set $i = i + 1$, go to step (2).

(5) Set iteration number $j = 0$, the initial exponent residual $\beta_0 = T_I$, initial vector $(x'_0, y'_0) = (x_I, y_I)$, to be compensated by scale factor.

(6) Check leading-bit position k and obtain $\beta_{j,s}$ of β_j. If $\beta_j \neq 0$, go to step (7). Otherwise $\beta_j = 0$: decompositions and compensations of variable factors are completed; let $J = j - 1$ and output the final rotated vector $(x', y') = (x'_J, y'_J)$.

(7) Using $\beta_{j,s}$, retrieve the optimized ρ_{l_j} and ρ_{m_j} information, $\ln(1+\rho_{l_j}2^{-l_j})$ and $\ln(1+\rho_{m_j}2^{-m_j})$ from lookup tables, and perform the iteration:

$$\begin{aligned} x'_{j+1} &= x'_j\left(1+\rho_{l_j}2^{-l_j}\right)\left(1+\rho_{m_j}2^{-m_j}\right) \\ y'_{j+1} &= y'_j\left(1+\rho_{l_j}2^{-l_j}\right)\left(1+\rho_{m_j}2^{-m_j}\right) \end{aligned} \quad (3.8)$$

$$\beta_{i+1} = \beta_i - \left[\ln\left(1+\rho_{l_j}2^{-l_j}\right) + \ln(1+\rho_m 2^{-m_j})\right] \quad (3.9)$$

(8) Set $j = j + 1$ and go to step (6).

Figure 1 shows the architecture for the rotation operations and exponent accumulation of the new CORDIC processor, while Fig. 2 is the architecture for scale

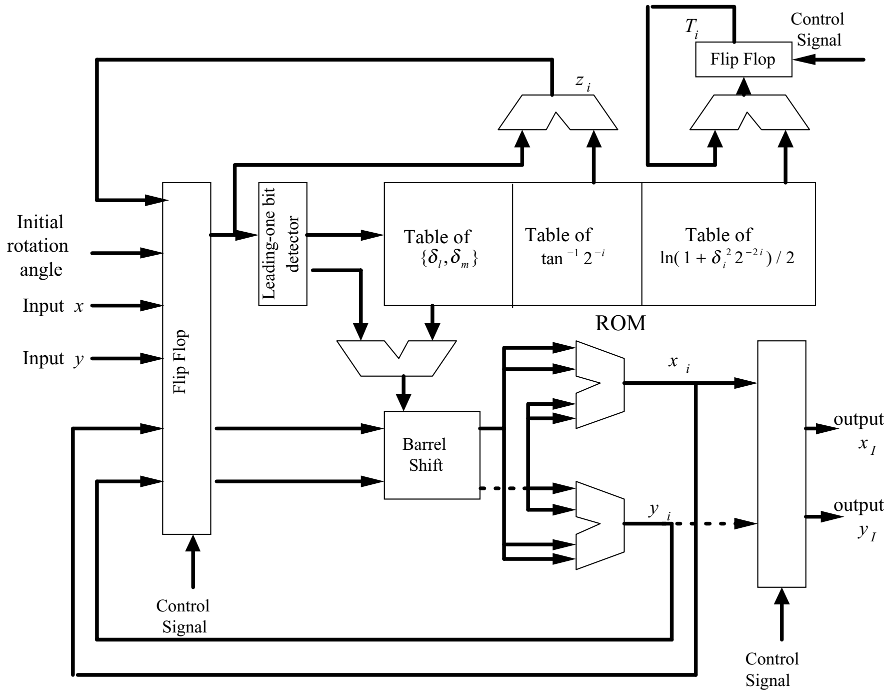


Figure 1. Structure of the new CORDIC algorithm (for rotation and exponent accumulation).

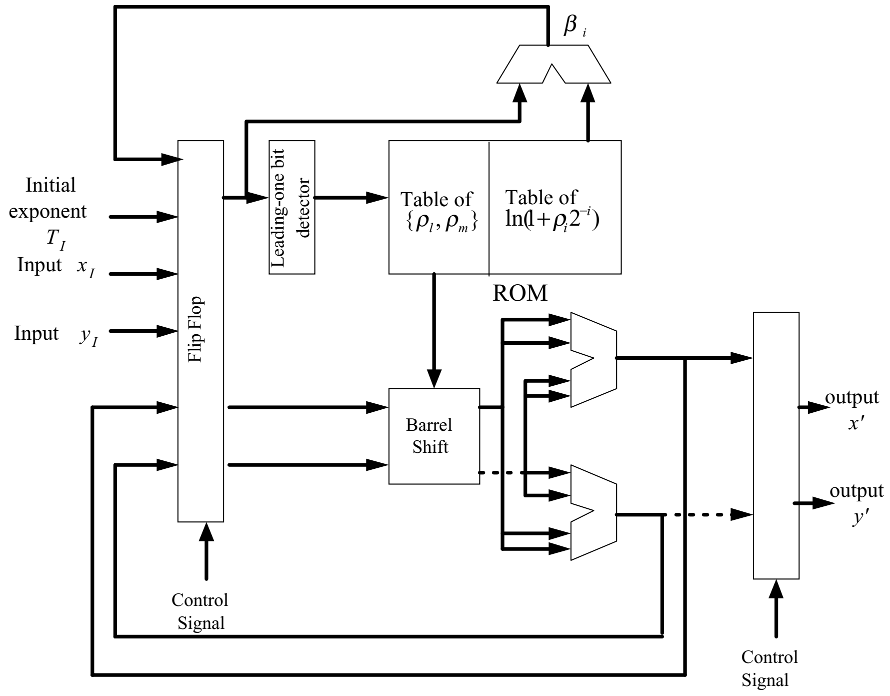


Figure 2. Structure of the new CORDIC algorithm (for scale factor decomposition and compensation).

decompositions and compensations. These two architectures can be merged as one, because their operations are similar. However, for the consideration of high-speed operations, they can be put in a pipelined structure in cascade. The pipelined structure is particularly efficient for the applications that require intensive and sustaining vector rotation operations.

4. Evaluation and Simulation

First, let's discuss the construction of the recoding tables for residual rotation angles. For the ease of realization, we only consider the recoding table that will produce positive residual rotation angles. As such, we don't have to consider the detection of leading-zero bit for negative residuals. As mentioned before, for best convergence performance, each possible value k of the leading-one bit position of the residual rotation angle needs an optimized table of its own. That means the total table size (in bits) required by the new optimized CORDIC algorithm is

$$S_t = \left[\sum_{k=0}^{n-1} (2 \times q_k \times 2^{r-1}) + n \times p \right] + \frac{n}{2} \times p' + \left[\sum_{k=0}^{n-1} (2 \times q'_k \times 2^{r-1}) + n \times p' \right] \tag{4.1}$$

where $\sum_{k=0}^{n-1} (2 \times q_k \times 2^{r-1})$: table size for the recoding of residual rotation angles; q_k: range (number) of δ_i's of the k-th recoding table; $n \times p$: table size for $\tan^{-1} 2^{-i}$'s, required by general CORDIC algorithms; $\frac{n}{2} \times p'$: table size for $(-\frac{1}{2}) \ln(1 + 2^{-2i})$, in exponent accumulation; $\sum_{k=0}^{n-1} (2 \times q'_k \times 2^{r-1})$: table size for the recoding of variable scale factors; q'_k: range (number) of ρ_i's of the k-th recoding table; $n \times p'$: table size for $\ln(1 + \rho_i 2^{-i})$, in scale factor decomposition; p': word length used in scale factor decomposition.

It seems that (4.1) introduces considerable memory overhead. However, the table size $\sum_{k=0}^{n-1}(2 \times q_k \times 2^{r-1})$ can be reduced to about $\sum_{k=0}^{n/3-1}(2 \times q_k \times 2^{r-1})$ by knowing that $\tan^{-1} 2^{-i} \approx 2^{-i}$, for $i > n/3$. So is $\sum_{k=0}^{n-1} (2 \times q'_k \times 2^{r-1})$. Moreover, r and q_k are generally much smaller than n. More significantly, after intensive simulations, we found that two recoding tables (corresponding to $k = \{0, 1\}$ and $k > 1$, respectively) for residual rotation angles, and one recoding table for the residual exponents, are enough to generate comparable performance to that of the optimized case. Doing this way, the total table size is reduced to

$$S_t = [2 \times (q_0 + q_1) \times 2^{r-1} + n \times p] + \frac{n}{2} \times p' + [2 \times q'_0 \times 2^{r-1} + n \times p'] \tag{4.2}$$

In the equation, obviously the original table size term np and the new term np' are the dominant terms. Also note that the values of $T_i = \ln(1+\delta_i^2 2^{-2i})/2$ and $a_i = \ln(1+2^{-i})$ reduce to zero and 2^{-i}, respectively, when roughly $i > n/2$. These properties can be utilized to reduce table size. Therefore, the total table size is roughly two to three folds that of the general CORDIC algorithms.

Based on the structures shown in Figs. 1 and 2, we performed fixed-point hardware simulations using Verilog hardware description language, assuming 22-bit accuracy (including 6-bit integer part and 16-bit fractional part). Exhausted simulations were conducted for all the rotation angles in the range of $0°–45°$. The total simulation patterns are 2949120, which are from 000000.000...000 to 101101.000......000. Table 4 shows and compares the simulation statistics of the sub-optimized case (with three recoding tables) and the

Table 4. Statistics of iteration numbers for the new optimized and sub-optimized CORDIC algorithms.

Total rotation number	Number of occurrences	
	Sub-optimized case	Optimized case
00	1	1
01	256	510
02	5940	11924
03	16096	35313
04	4278	31010
05	17952	48210
06	35228	48298
07	62292	114536
08	95766	162424
09	128265	295286
10	241874	601588
11	404136	1104256
12	1062108	413326
13	741469	78755
14	123881	3683
15	9578	0
Total	2949120	2949120
Average	10.74	9.78

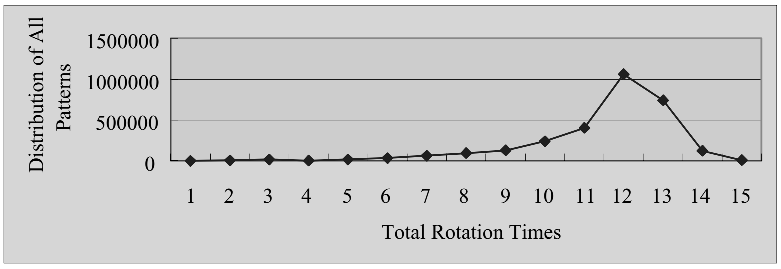


Figure 3. Iteration number distribution of the new sub-optimized CORDIC algorithm.

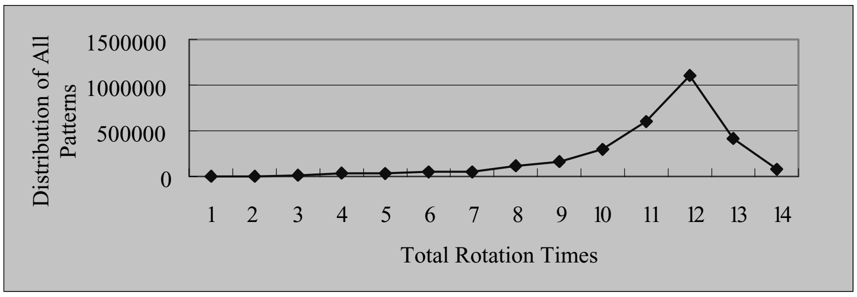


Figure 4. Iteration number distribution of the new optimized CORDIC algorithm.

optimized case (with 12 recoding tables). Figure 3 is the iteration distribution graph corresponds to the sub-optimal case, while Fig. 4 corresponds to the optimized case (through computer search). The sub-optimal case results in an averaged iteration number of 10.74, which is a little larger than the optimized design with an averaged iteration number of 9.78. This small 10% increase in iteration number has the advantage of a much reduced table size, in comparison to the optimized case. Note that for both cases, we use the same condition of $q_k = 8$, for $k = 0, 1$, $q_k = 7$, for $k > 1$, and $q'_k = 5$, which are also optimized by computer search.

It can be argued statistically, that total iteration number required by the new algorithm is less than $n/2$, for the whole operations of rotation and scale factor decomposition and compensation, where n is the target data precision in bits. Although the new design has to perform 4-operand addition operations per iteration, it can be easily reduced to 2-operand additions by using the low-cost and fast carry-save adders. Comparison of the iteration numbers, iteration latencies and area performances of the new design with some of the comparable efficient designs [12, 14, 16] is shown in Table 5, for the realization of serial computation. As mentioned in the introduction, although the very high-radix CORDIC algorithm [20, 21] enjoys very small iteration numbers, it needs two complicated MAC units with a long iteration time. In implementation, its efficiency is highly dependent on circuit designers' expertise. Therefore, we will not include it in the comparison table. The CORDIC algorithm with close-to-optimal angle recoding scheme [11] can reduce the iteration number to $n/3$ in average (excluding penalty iterations for the introduced complicated variable scale factors). However, it has to perform $O(n^2)$ comparison operations. That is a huge overhead compared to the other CORDIC algorithms. The differential CORDIC algorithm [18] is designed for efficient parallel pipeline operations. It is not suited for serial computation, due to its long initial delay of n time units. In addition, it needs n iterations for micro-rotations plus $O(n)$ shift-and-add iterations for constant scale factor compensation.

In the comparison table, the fast prediction algorithm [16], the radix-4 algorithms [12, 14], all have a low iteration number of about $0.8n$ (including micro-rotations and variable scale factor compensations). Normally the dominant timing terms (in descending order) contributing to the iteration latency are: delay time t_{bs} of barrel shifter, and adder delay time t_{ad}. These two terms generally are significantly greater the other terms such as t_{ts}, t_{mux}, t_{fa}, t_{ld}, t_{mbe} and $t_{ad'}$ (all are defined beneath Table 5). In order to roughly quantify the comparison,

Table 5. Performance comparison of competing CORDIC's.

		Iteration latency (excluding t_{ts})			
Algorithm	Iteration number	$n = 16$	$n = 32$	$n = 64$	Extra/Area (w.r.t conventional algorithm)
Conventional algorithm	$4n/3$	$(t_{bs} + t_{ad}) + t_{ts}$			0
		1	1	1	
Fast-predict algorithm [16]	$4n/5 + \log_3(n)$	$(t_{bs} + t_{ad}) + t_{ts} + t_{mbe} + t_{mux}$			$A_{mbe} + 3n^2/2$ bit cells
		1.30	1.20	1.13	
Radix-4 algorithm [12]	$4n/5$	$(t_{ld} + t_{bs} + t_{ad}) + t_{ts}$			$A_{ld} + 3n^2/2$ bit cells
		1.13	1.10	1.06	
Radix-4 algorithm [14]	$4n/5$	$(t_{ld} + t_{bs} + t_{ad}) + t_{ts} + 2t_{mux}$			$A_{bs} + A_{ad} + A_{ld} + 3^{n/12+1}n$ bit cells
		1.36	1.25	1.17	
New algorithm	Less than $n/2$	$(t_{ld} + t_{ad} + t_{bs}) + t_{ts} + 2t_{fa} + t_{ad'}$			$A_{ld} + 2nA_{fa} + (3n^2/2 + 16 \times 8)$ bit cells
		1.65	1.47	1.32	

Where t_{ts} is the delay time for table selection, t_{bs} is the delay time of barrel shifter, t_{ad} is the delay time of an n-bit adder, t_{ld} is the delay time for leading-one bit detection, which is the same as the delay time for zero-skipping, t_{mux} is the delay time of a MUX circuit, t_{mbe} is the delay time for a modified Booth encoder, $t_{ad'}$ is the delay time of a $\log_2 n$-bit adder, t_{fa} is the delay time of an one-bit full adder, A_t is the size of the lookup table as detailed in Table 7, A_{bs} is the size of an n-bit barrel shifter, A_{ad} is the size of an n-bit adder, A_{reg} is the size of an one-bit register, A_{ld} is the size of a leading-one bit detector, which is the same as that of a zero-skipping circuit, A_{mbe} is the size of a modified Booth encoder and A_{fa} is the size of an one-bit full adder.

Table 6. Typical delays and areas of key cells.

Key cell delay/area	$n = 16$ bits	$n = 32$ bits	$n = 64$ bits
t_{bs}/A_{bs}	2.67 ns (2234 μm^2)	4.46 ns (5740 μm^2)	6.87 ns (13797 μm^2)
t_{ad}/A_{ad}	1.53 ns (11053 μm^2)	1.83 ns (25239 μm^2)	2.72 ns (41015 μm^2)
$t_{ad'}/A_{ad'}$	0.81 ns (2349 μm^2)	0.96 ns (2859 μm^2)	1.05 ns (3283 μm^2)
t_{ld}/A_{ld}	0.54 ns (976 μm^2)	0.60 ns (1389 μm^2)	0.65 ns (3820 μm^2)
t_{mbe}/A_{mbe}		0.78 ns (652 μm^2)	
t_{fa}/A_{fa}		0.69 ns (150 μm^2)	
t_{mux}		0.49 ns (46 μm^2)	

we synthesized most of the key circuit modules in the critical path for each design, by using Synopsys synthesis tool and UMC 0.25 μm standard cell, assuming a maximum transition time 0.8 ns (suggested by the cell library) and a maximum 10-gate fanout. The synthesis results are shown in Table 6. Of course, the simulations results only reflect general characteristics.

As shown, the simulation results confirm the previous argument on delay time. We can see that the delay sum $t_{ad} + t_{bs} + t_{ts}$ dominates the critical path. This sum term is required by every design, which is significantly larger than the sum of all the other timing terms. The numbers shown in Table 5 are the normalized iteration latencies with respect to the iteration latency of the conventional CORDIC structure. Although the new design's delay time is the longest among all, it is only 1.32 to 1.65 times that of the conventional one. This delay time is much closer to the latencies of the other designs. Note that these figures do not take t_{ts}, setup time and hold time into account, which are required in realization. If they are included in the calculation, then all the normalized latencies will be closer to one than before. Compared to the conventional CORDIC algorithms, the greatly reduced iteration number (from $4n/3$ to $n/2$) of the new design is more than enough to compensate the slightly lengthened iteration latency.

Based on the area costs given in Table 5, it is hard to compare the area performances precisely, especially when they are in the same order of magnitude. Under this condition, the actual area costs are highly dependent on process, design methodology, cell library, synthesis tool and circuit designers' expertise.

Table 7. Extra ROM table sizes in addition to the $\tan^{-1} 2^{-i}$ table.

Algorithm	A_t (no. of bit cells)
Conventional algorithm	$n \times n$
Fast-predict algorithm [16]	$5n/2 \times n$
Radix-4 algorithm [12]	$5n/2 \times n$
Radix-4 algorithm [14]	$(n + 3^{n/12+1}) \times n$
New Algorithm (for radix-16)	$(5n/2) \times n + 16 \times 8$

Therefore, we only give a brief discussion as follows. Table 5 lists other area costs than the common term $(2A_{bs} + 3A_{ad}) + A_t + 3nA_{reg}$ (corresponding to the required area of the conventional CORDIC). This common quantity is the most dominant area contributor, because it consists of the relatively large barrel shifters and adders. Compared to the conventional CORDIC, all the other designs in Table 5 need extra ROM table areas. Typically, a ROM bit cell only occupies an area of about three transistors. The extra ROM table sizes listed in Table 7 are estimated to be noticeably smaller than the common area term. Note that the extra term $2nA_{fa}$ of the new design corresponds to an area of $2n$ full adders, which is only a small portion of total area. Hence, it is reasonable to conclude that the new design has a total area less than two times that of the conventional CORDIC, and its area is comparable to that of the radix-4 CORDIC of [14].

5. Conclusion

The proposed new CORDIC algorithm considerably reduces iteration number. It is achieved by combining several design techniques. The new algorithm is generic enough to be extended to higher radices for lower iteration numbers, but at larger ROM sizes. The vectoring mode CORDIC and hyper-trigonometric CORDIC operations using those design techniques are not addressed here, which will be reported in the future work. Similarly, the new recoding scheme can also be applied to other basic functions such as the division and square-root operations.

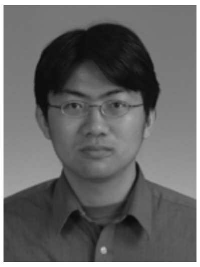
Acknowledgment

This work is supported in part by the National Science Council, Taiwan, under contract no. NSC 89-2218-E-009-079.

References

1. J.E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. Electronic Comput.,* vol. EC-8, 1959, pp. 330–334.
2. J.S. Walther, "A Unified Algorithm for Elementary Functions," in *AFIPS Spring Joint Comput. Conf.*, 1971, pp. 379–385.
3. N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC Methods with a Scale Factor for Sine and Cosine Computation," *IEEE Trans. on computers,* vol. 40, no. 9, 1991, pp. 989–995.
4. J. Duprat and J.M. Muller, "The CORDIC Algorithm: New Results for Fast VLSI Implementation," *IEEE Trans. on Comput.*, vol. 42, no. 2, 1993, pp. 168–178.
5. M. Kuhlmann and K.K. Parhi, "A High-Speed CORDIC Algorithm and Architecture for DSP Applications," in *SiPS 99, 1999 IEEE Workshop*, 1999, pp. 732–741.
6. D.S. Phatak, "Double Step Branching CORDIC: A New Algorithm for Fast Sine and Cosine Generation," *IEEE Trans. on Computer*, vol. 47, no. 5, 1998, pp. 587–602.
7. J.R. Cavallaro and N.D. Hemkumar, "Redundant and On-Line CORDIC for Unitary Transformations," *IEEE Trans. Comput.*, vol. 43, no. 8, 1994, pp. 941–954.
8. J.A. Lee and T. Lang, "Constant-Factor Redundant CORDIC for Angle Calculation and Rotation," *IEEE Trans. Comput.*, vol. 41, no. 8, 1992, pp. 1016–1025.
9. M.D. Ercegovac and T. Lang, "Redundant and On-Line CORDIC: Application to Matrix Triangularization and SVD," *IEEE Trans. Comput.*, vol. 39, no. 6, 1990, pp. 725–740.
10. H.X. Lin and H.J. Sips, "On-line CORDIC Algorithms," *IEEE Trans. Comput.*, vol. 39, no. 8, 1990, pp. 1038–1052.
11. Y.H. Hu and S. Naganathan, "An Angle Recoding Method for CORDIC Algorithm Implementation," *IEEE Trans. on Computers*, vol. 42, no. 1, 1993, pp. 99–102.
12. C.C. Li and S.G. Chen, "A Radix-4 Redundant CORDIC Algorithm with Fast On-Line Variable Scale Factor Compensation," in *Proc. of 1997 IEEE International Conference on Acoustic, Speech and Signal Processing*, Munich 1997, Germany, pp. 639–642.
13. M.R.D. Rodrigues, "Hardware Evaluation of Mathematical Functions," *IEE Proc.*, vol. 128, pt. E, no. 4, 1981.
14. E. Antelo, J. Villalba, D. Bruguera, and E.L. Zapata, "High Performance Rotation Architectures Based on the Radix-4 CORDIC Algorithm," *IEEE Trans. on Computers*, vol. 46, no. 8, 1997, pp. 855–870.
15. D. Timmermann, H. Hahn, and B.J. Hosticka, "Low Latency Time CORDIC Algorithms," *IEEE Trans. on Computers*, vol. 41, no. 8, 1992, pp. 1010–1015.
16. S.G. Chen and C.F. Lin, "A CORDIC Algorithm with Fast Rotation Prediction and Small Iteration Number," in *Proceedings of 1998 IEEE International Symposium on Circuits and Systems, 1998*, Monterey, CA, USA, pp. V229–V232.
17. J.H. Kwak, J.H. Choi, and E.E. Swartzlander, Jr. "High-Speed CORDIC Based on an Overlapped Architecture and a Novel σ-Prediction Method,"*Journal of VLSI Signal Processing*, vol. 25, 2000, pp. 167–177.
18. H. Dawid and H. Meyr, "The Differential CORDIC Algorithm: Constant Scale Factor Redundant Implementation without Correcting Iterations," *IEEE Trans. on Computers*, vol. 45, no. 3, 1996, pp. 307–318.

19. C.C. Li and S.G. Chen, “New Redundant CORDIC Algorithm with Variable Scale Factor Compensations,” in *Proceedings of 1996 IEEE International Symposium Circuits and Systems*, May 1996, Atlanta, USA, pp. 264–267.
20. E. Antelo, T. Lang, and J.D. Bruguera, “Very-High Radix CORDIC Rotation Based on Selection by Rounding,”*Journal of VLSI Signal Processing*, vol. 25, 2000, pp. 141–153.
21. E. Antelo, T. Lang, and J.D. Bruguera, “Very-High Radix Circular CORDIC: Vectoring and Unified Rotation/Vectoring,” *IEEE Trans. on Computer*, vol. 49, no. 7, 2000, pp. 727–739.
22. T.C. Chen, “Automatic Computation of Exponentials, Logarithms, Ratios and Square Roots,” *IBM Journal Res. and Dev.*, vol. 16, 1972, pp. 380–388.



Jen-Chuan Chih was born in Taipei, R.O.C. He received the B.S. and M.S. degrees in electrical engineering from the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Taiwan in 1998 and 1999, respectively. Currently he is pursuing the Ph.D. degree. His research interests are in fast DSP algorithms and VLSI implementation.

Sau-Gee Chen received the B.S. degree in nuclear engineering from the National Tsing Hua University, Taiwan, in 1978, the M.S. degree and Ph.D. degree in electrical engineering, from the State University of New York at Buffalo, NY, in 1984 and 1988 respectively. He currently is a Professor of the Department Engineering, National Chiao Tung University, Taiwan. His research interests include multimedia computing, digital signal processing, and VLSI circuit design.
sgchen@cc.nctu.edu.tw