A Unified View for Vector Rotational CORDIC Algorithms and Architectures Based on Angle Quantization Approach

An-Yeu Wu and Cheng-Shing Wu

***Abstract*—Vector rotation is the key operation employed extensively in many digital signal processing applications. In this paper, we introduce a new design concept called *Angle Quantization (AQ)*. It can be used as a design index for vector rotational operation, where the rotational angle is known in advance. Based on the AQ process, we establish a unified design framework for cost-effective low-latency rotational algorithms and architectures. Several existing works, such as conventional *COordinate Rotational DIgital Computer* (CORDIC), AR-CORDIC, MVR-CORDIC, and EEAS-based CORDIC, can be fitted into the design framework, forming a *Vector Rotational CORDIC Family*. Moreover, we address four searching algorithms to solve the optimization problem encountered in the proposed vector rotational CORDIC family. The corresponding scaling operations of the CORDIC family are also discussed. Based on the new design framework, we can realize high-speed/low-complexity rotational VLSI circuits, whereas without degrading the precision performance in fixed-point implementations.**

***Index Terms*—Angle Quantization (AQ), Angle Recoding (AR), greedy searching algorithm, trellis-based searching (TBS) algorithm, vector rotational COordinate Rotational DIgital Computer (CORDIC) algorithm.**

I. INTRODUCTION

VECTOR rotation plays an important role in many digital signal processing (DSP) applications. It is extensively employed as the processing kernel in discrete orthogonal transformations (DCT, DST, and DFT) [1]–[4], lattice-based (rotation-based) digital filtering [5], [6], sinewave/cosine generation [7], [8], and digital modulation/demodulation in communication systems [9], [10]. Let $[x_{in} \;\; y_{in}]^T$ and $[x_{out} \;\; y_{out}]^T$ denote the input and output vectors, respectively. Vector rotation of $[x_{in} \; y_{in}]^T$ by a rotational angle θ can be formulated as

$$\begin{bmatrix} x_{\text{out}} \\ y_{\text{out}} \end{bmatrix} = \mathbf{R} \cdot \begin{bmatrix} x_{\text{in}} \\ y_{\text{in}} \end{bmatrix} \tag{1}$$

where the *rotational matrix* $\mathbf{R}$ is defined as

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{2}$$

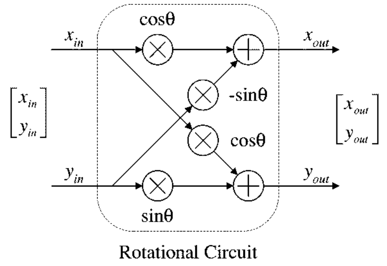
cosθ
x_{in}
x_{out}
$\begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$
-sinθ
cosθ
$\begin{bmatrix} x_{out} \\ y_{out} \end{bmatrix}$
y_{in}
y_{out}
sinθ
Rotational Circuit

Fig. 1. Direct implementation of rotational circuit.

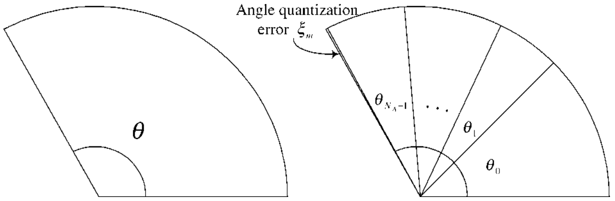

Fig. 2. Concept of *Angle Quantization*, where $\theta = (\theta_0 + \theta_1 + \cdots + \theta_{N_A-1}) + \xi_m$.

Fig. 1 shows the direct implementation of (2). As one can see, the direct implementation calls for four multipliers and two adders, and the critical path is the total delay of one multiplier and one adder. The direct implementation is very area-consuming and low-speed when rotational operations are heavily utilized in VLSI circuits.

In practical implementations, to increase the operational speed of rotation, one effective way is to employ high-speed arithmetic operators, such as *Carry-lookahead Adder* and *Recoded Multipliers* [11], i.e., to reduce the delay of critical path. In general, the speed benefit is gained at the expense of advanced arithmetic operators. We call them *Arithmetic-based approaches*.

On the other hand, we can achieve high-speed/low-complexity rotational circuits by *sacrificing rotation precision* in fixed-point implementations. For example, the hardware complexity can be significantly reduced as the assigned wordlength is shortened, hence the critical path. Also, *Signed Power-of-Two* (SPT) [12], [13] coding of coefficient parameters ($\sin\theta$ and $\cos\theta$) is an alternative way to reduce the complexity of the direct implementation. Both approaches introduce quantization error due to the *approximation process* of rotational matrix $\mathbf{R}$, which causes degradation of the signal quality in practical implementations. The amount of quantization error depends on the

---

Manuscript received July 25, 2001; revised April 8, 2002. This work was supported in part by the National Science Council, R.O.C., under Grant NSC 89-2218-E-002-108. This paper was recommended by Associate Editor W. B. Mikhael.

A.-Y. Wu is with the Graduate Institute of Electronics Engineering and the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C.

C.-S. Wu is with the Department of Electrical Engineering, National Central University, Chung-Li 320, Taiwan, R.O.C.

Digital Object Identifier 10.1109/TCSI.2002.803363.

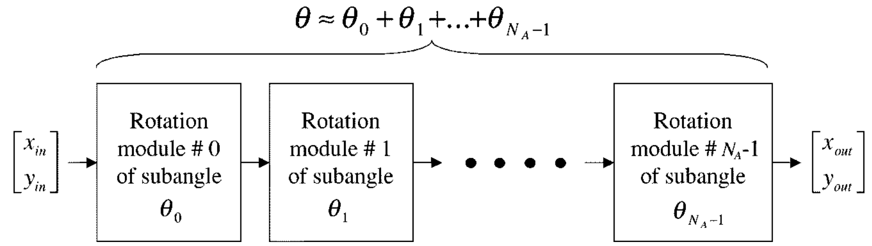
$\theta \approx \theta_0 + \theta_1 + ... + \theta_{N_A - 1}$
x_{in}
y_{in}
Rotation module # 0 of subangle θ_0
Rotation module # 1 of subangle θ_1
Rotation module # N_A-1 of subangle $\theta_{N_A - 1}$
x_{out}
y_{out}

Fig. 3. Realization of fast vector rotation operation based on the AQ process.

assigned wordlength and number of nonzero digits employed. We call such schemes *Approximation-based approaches*.

Following the concept of approximation-based approaches, in this paper, we propose a novel framework to design high-speed/low-cost vector rotational VLSI circuits. Instead of performing quantization on the coefficient parameters ($\cos\theta$ and $\sin\theta$), the proposed design framework originates from the concept of *Angle Quantization (AQ)*. The AQ derives the name from the fact that we perform the quantization process on the rotational angle, θ, directly. That is, we *decompose* the original rotational angle θ into several *sub-angles*, θ_is. Then, we try to sum up those sub-angles to approximate the original angle as close as possible; or equivalently, we try to minimize the *angle quantization error*

$$\xi_m \triangleq \theta - \sum_{i=0}^{N_A - 1} \theta_i \tag{3}$$

where N_A denotes the number of sub-angles. The AQ process is demonstrated in Fig. 2. Based on the AQ process, The vector rotation operation can be realized as shown in Fig. 3. Each rotation module is dedicated to performing a particular rotation of sub-angle θ_i. Then, the rotation of θ can be accomplished by cascading these N_A rotation modules.

In the AQ process, there are two key design issues:

1) Firstly, we need to *determine (or construct)* the sub-angles, and each θ_i needs to be easy-to-implement in practical VLSI circuits.
2) Secondly, we have to find out how to *select and combine* these sub-angles such that the angle quantization error ξ_m can be suppressed.

In fact, the well-known *COordinate Rotational DIgital Computer (CORDIC)* algorithm [14]–[16] can be considered as an approach to perform the AQ process. Recall that in the CORDIC algorithm, the rotation of angle θ is performed by *sequentially* rotating elementary angle of $a(i) = \tan^{-1}(2^{-i})$, for $0 \leq i \leq W - 1$, where W denotes the wordlength. The advantageous feature of the elementary angle is that rotation of $a(i)$ requires only two shift-and-add operators. The easy-to-implement feature of $a(i)$ conforms to the requirements of aforementioned AQ process. In addition, the sequential rotating operation of $a(i)$s is the way to select and combine those sub-angles in conventional CORDIC.

Next, we can link the AQ process with several existing vector rotation schemes, such as Angle Recoding (AR) technique [17], Modified Vector Rotational CORDIC (MVR-CORDIC) algorithm [18] and Extended Elementary Angle Set (EEAS) scheme [19]. We explore their relationship with the proposed AQ process. Then we will derive a unified framework for all these vector rotational operations. That is, all previous schemes can be considered as subsets of the proposed framework. The unified operations and AQ process of these algorithm suggest a family of rotation algorithms. We call it *Vector Rotational CORDIC Family*.

The rest of the paper is organized as follows. In Section II, we discusses the basic concept of angle quantization. Then we explore the relationship between AQ process and the conventional CORDIC algorithm. In Section III, we address several existing vector rotation schemes as well as their relationship with the proposed AQ process. Then we derive the proposed vector rotational framework. In Section IV, we discuss four searching algorithms to solve the optimization problem encountered in AQ process. The searching algorithms help to select and combine the aforementioned sub-angles so as to achieve the desired precision performance. In Section V, we focus on the scaling operation of the vector rotational CORDIC family. In Sections VI and VII, the performance comparisons and VLSI architectures of the Vector Rotational CORDIC Family are addressed, respectively. Finally, we conclude our work in Section VIII.

II. ANGLE QUANTIZATION AND CONVENTIONAL CORDIC ALGORITHM

A. Angle Quantization Process and CSD

Angle Quantization (AQ) is conceptually similar to the *Canonical Signed Digit (CSD)* Quantization scheme [11]–[13]. In many digital filter designs, to save hardware complexity, the coefficient is *recoded* in the form of sum of *Signed-Power-of-Two (SPT) terms*. For example, coefficient of $h_2 = (-0.156\,249)_{10}$ can be recoded as $h_2 = (0.0\bar{1}011)_2$ for wordlength $W = 8$ with three nonzero digits, where $\bar{1}$ represents -1. With such manipulations, multiplication can be easily realized with only three *shift-and-add* operators, as shown in Fig. 4. Hence, the hardware complexity and the critical path can be significantly reduced.

Conceptually, the CSD approach is performed through the following two steps.

1) Firstly, CSD quantization *decomposes* the coefficient into several SPT terms, i.e., sub-coefficients. The operation of each sub-coefficient can be easily realized.
2) Secondly, the multiplication of h_i can be reformed through the combination of these SPT sub-coefficients.

By following the concept of CSD, we quantize the rotation angle θ as demonstrated in Fig. 2. We first decompose the rotation angle θ into several *sub-angles*, θ_is. Next, similar to aforementioned CSD scheme, the design criteria of all sub-angles,

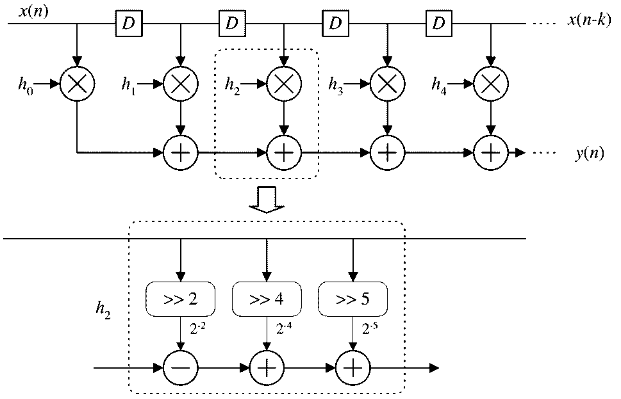

Fig. 4. Illustration of CSD quantization of filter coefficients.

TABLE I

SUMMARY OF CORRESPONDENCES BETWEEN CSD AND ANGLE QUANTIZATION, WHERE $g_j \in \{-1, 0, 1\}$, $d_j \in \{0, 1, \ldots, W-1\}$, N_D AND N_A DENOTE THE NUMBERS OF NON-ZERO DIGITS AND SUB-ANGLES, RESPECTIVELY

	Canonical Sign Digit (CSD)	Angle Quantization (AQ)
Approximation Target	Coefficient, h_k	Rotation angle, θ
Basic Element	Non-zero digit, 2^{-i}	Sub-angle, $a(i) = \tan^{-1}(2^{-i})$
Basic Operation	Shift-and-add operation	2 Shift-and-add operations
Approximation Equation	$h_k \approx \sum_{j=0}^{N_D-1} g_j \cdot 2^{-d_j}$	$\theta \approx \sum_{j=0}^{R_m-1} \alpha(j) \cdot a(s(j))$

θ_is, is that the rotational operation of each θ_i should be easily realized (just like the SPT terms in filter design). Suppose that each θ_i can also be realized using only shift-and-add operations. Then, with the help of easy-to-implement sub-angles, the rotation of θ can be performed through successive applications of sub-angle rotations in a cost-effective way. In Table I, we summarize the correspondences between the CSD quantization scheme and the proposed AQ process.

B. Operations of Conventional CORDIC Algorithm

In conventional CORDIC algorithm, the *elementary angles*, $a(i)$, is defined as [14]–[16]

$$a(i) \triangleq \tan^{-1}(2^{-i}). \tag{4}$$

By substituting $\theta = a(i) = \tan^{-1}(2^{-i})$ into (2), the ith micro-rotation of CORDIC can be represented as

$$\begin{aligned}\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} &= (1+2^{-2i})^{0.5} \begin{bmatrix} x_p(i+1) \\ y_p(i+1) \end{bmatrix} \\ &= \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}\end{aligned} \tag{5}$$

where $[x_p(i+1)\ y_p(i+1)]^T$ denotes the ideal rotated vector. The operation is demonstrated in Fig. 5. As we can see from (5), the rotation of $a(i)$ requires only two shift-and-add operators, which can be easy to realized in VLSI circuits. The easy-to-implement feature of $a(i)$ conforms to the requirement of sub-angles in AQ process.

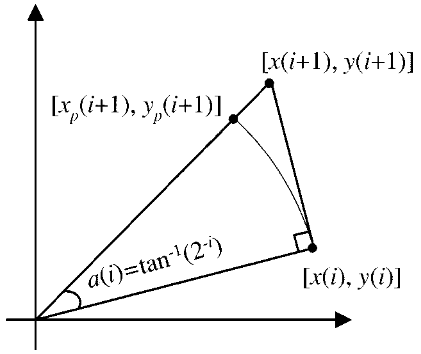

Fig. 5. Illustration of the ith micro-rotation in conventional CORDIC algorithm.

TABLE II

SUMMARY OF CORDIC ALGORITHM IN CIRCULAR ROTATIONAL MODE

```
% Initialization
Given $x(0), y(0)$, and $z(0)$
% Micro-rotation phase
FOR $i = 0$ to $N-1$
        $\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\mu(i)2^{-i} \\ \mu(i)2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$
        % Angle updating
        $z(i+1) = z(i) - \mu(i)a(i)$, where $a(i) = \tan^{-1}(2^{-i})$
END
% Scaling phase
$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = P \begin{bmatrix} x(N) \\ y(N) \end{bmatrix} = \left( \prod_{i=0}^{N-1} \sqrt{1+2^{-2i}} \right)^{-1} \begin{bmatrix} x(N) \\ y(N) \end{bmatrix}$
```

Based on the elementary angles, the conventional CORDIC algorithm can be rewritten as [14]–[16]

$$\theta = \sum_{i=0}^{N-1} \mu(i)a(i) + \xi_m \tag{6}$$

where N denotes the number of elementary angles, $\mu(i) \in \{1, -1\}$ is the *rotation sequence* which determines the ith rotational angle $a(i)$. In general, for data of W-bit wordlength, the iteration number is less than W, i.e., $N \leq W$. Basically, the CORDIC tries to decompose the rotation angle, θ, into the combination of $a(i)$, for $i = 0, 1, \ldots, N-1$. The *angle quantization error* of the CORDIC algorithm

$$\xi_{m,\,\mathrm{CORDIC}} \triangleq \theta - \left[\sum_{i=0}^{N-1} \mu(i)a(i) \right] \tag{7}$$

represents the *residue angle* beyond the resolution of CORDIC algorithm. In Table II, we summarize the basic iteration procedure of the CORDIC algorithm in the *circular mode*.

C. Link AQ Process With Conventional CORDIC Algorithm

Next, we would like to define *Elementary Angle Set (EAS)* for the derivation of the proposed vector rotational framework. Basically, EAS consists of all elementary angles used in the rotation algorithms. In the conventional CORDIC algorithm, the

TABLE III
SUMMARY OF THE ELEMENTARY ANGLES EMPLOYED IN CORDIC ALGORITHM
(ASSUME WORDLENGTH $W = 8$)

Iteration number	Elementary Angle	Value in Radius
$i = 0$	$a(0) = \tan^{-1}(2^{-0})$	0.78539816339745
$i = 1$	$a(1) = \tan^{-1}(2^{-1})$	0.46364760900081
$i = 2$	$a(2) = \tan^{-1}(2^{-2})$	0.24497866312686
$i = 3$	$a(3) = \tan^{-1}(2^{-3})$	0.12435499454676
$i = 4$	$a(4) = \tan^{-1}(2^{-4})$	0.06241880999596
$i = 5$	$a(5) = \tan^{-1}(2^{-5})$	0.03123983343027
$i = 6$	$a(6) = \tan^{-1}(2^{-6})$	0.01562372862048
$i = 7$	$a(7) = \tan^{-1}(2^{-7})$	0.00781234106010

EAS comprises of all $a(i)$, for $0 \le i \le N-1$, and can be defined as

$$\mathcal{S} = \{a(i)\colon 0 \le i \le N-1\}. \tag{8}$$

In Table III, we illustrate the EAS of the CORDIC with $W = 8$.

With the help of EAS, we can say that the CORDIC algorithm essentially performs the angle quantization. This can be observed from (6). Given a target rotation angle θ, CORDIC algorithm determines the first rotation sequence $\mu(0)$ for the most significant elementary angle $a(0)$ (the topmost elementary angle in Table III), followed by the determination of $\mu(1)$ for $a(1)$. The process is repeated until the last elementary angle is applied. That is, the CORDIC algorithm tries to perform the rotation through *sequentially* applying micro-rotations of *all* elementary angles.

Referring to Table I, now we can relate AQ to CORDIC algorithm as follows:

1) The sub-angle θ_i in AQ now becomes

$$\theta_i = \mu(i)a(i) \tag{9}$$

in CORDIC algorithm.
2) The number of sub-angles of N_A in AQ is set to be N in CORDIC algorithm.
3) CORDIC algorithm *sequentially* apply *all* θ_i, for $i = 0, 1, \ldots, N-1$, to approximate the target angle θ.

III. DESIGN FRAMEWORK FOR VECTOR ROTATIONAL OPERATIONS

In previous section, we have shown that the CORDIC algorithm tries to perform AQ based on limited θ_is and fixed θ_i sequence. However, in the applications where the rotation angles are known in advance, it would be advantageous to relax the sequential constraint on the sub-angles (or micro-rotations), or to extend the EAS range. Based on the idea, a new design framework can be developed, and several previous designs of [17]–[19] can be fitted into this design framework.

A. AR Technique [17]

In conventional CORDIC algorithm, the micro-rotations of all elementary angles are performed in a sequential way. On the contrary, in the *Angle Recoding (AR)* technique proposed by Hu and Naganathan [17], certain micro-rotations can be *skipped* depending on the target rotational angle. Specifically, the modification is done by extending the set of $\mu(i)$ from $\{1, -1\}$ to $\{1, -1, 0\}$. By substituting $\mu(i) = 0$ into the recurrence equation in Table II, we obtain

$$\begin{cases} x(i+1) = x(i) \\ y(i+1) = y(i) \end{cases}. \tag{10}$$

Equation (10) represents a *null operation*, which can be ignored in practical implementation. Hence, by setting $\mu(i) = 0$, one can skip the micro-rotation of the elementary angle $a(i) = \tan^{-1}(2^{-i})$. This can help to reduce the iteration number, implying the speedup of CORDIC algorithm.

With the modification of (10), now the angle quantization error of the AR technique, $\xi_{m,AR}$, can be represented as

$$\xi_{m,\,AR} \triangleq \theta - \left[\sum_{i=0}^{N-1} \mu(i)a(i)\right]. \tag{11}$$

Basically, (11) is identical to (7), except for the extended $\mu(i) \in \{1, -1, 0\}$.

In practice, for certain target rotational angle θ, the AR technique can not only reduce the iteration number, but also suppress the angle quantization error. Take one extreme case for example. Consider the target angle of $\theta = \pi/4$. The conventional CORDIC approach has to go through all N micro-rotations with $\overline{\mu} = [1, -1, 1, 1, 1, \ldots]$, where $\overline{\mu}$ denotes the set $[\mu(0), \mu(1), \ldots \mu(N-1)]$. The angle quantization error of $\xi_{m,\mathrm{CORDIC}} = 7.2 * 10^{-3}$ for $N = 8$. However, with the AR technique, we can rotate $\theta = \pi/4$ by performing only one micro-rotation of elementary angle $a(0) = \tan^{-1}(2^{\circ})$, while skipping all remaining micro-rotations. The resultant rotation sequence is $\overline{\mu} = [1, 0, 0, 0, 0, \ldots]$, and it takes only one iteration with $\xi_{m,\,AR} = 0$.

In addition to minimizing $\xi_{m,\,AR}$, it would be desirable to minimize the *effective* iteration number $N' = \sum_{i=0}^{N-1} |\mu(i)|$. Hence, the iteration stages in implementation can be reduced. In summary, the AR optimization problem can be stated as:

> Given a rotation angle θ, find rotation sequence $\mu(i) \in \{-1, 0, 1\}$ for $0 \le i \le N-1$, such that the angle quantization error $|\xi_{m,\,AR}| < a(N-1)$, and the effective iteration number N' is minimized.

In [17], the *Greedy algorithm* is proposed to solve the optimization problem. We will discuss the issue in Section IV.

1) AQ Process in the AR Technique:

Elementary Angle Set (EAS)*:* To make AR technique fit into our design framework, we reformulate (11) of AR technique by removing the null operations of $\mu(i) = 0$ and applying the equality of $\pm\tan^{-1}(A) = \tan^{-1}(\pm A)$. After changing the variables and index, we can rewrite (11) in a compact form as

$$\xi_{m,\,AR} = \theta - \left[\sum_{j=0}^{N'-1} \tan^{-1}\left(\alpha(j)\cdot 2^{-s(j)}\right)\right] \triangleq \theta - \left[\sum_{j=0}^{N'-1} \tilde{\theta}(j)\right] \tag{12}$$

TABLE IV
SUMMARY OF THE EAS $\mathcal{S}_1$ EMPLOYED IN THE AR TECHNIQUE (ASSUME WORDLENGTH $W = 8$)

Elementary Angle	Value in Radius	Elementary Angle	Value in Radius
$r(1) = \tan^{-1}(-2^{-0})$	-0.78539816339745	$r(10) = \tan^{-1}(2^{-7})$	0.00781234106010
$r(2) = \tan^{-1}(-2^{-1})$	-0.46364760900081	$r(11) = \tan^{-1}(2^{-6})$	0.01562372862048
$r(3) = \tan^{-1}(-2^{-2})$	-0.24497866312686	$r(12) = \tan^{-1}(2^{-5})$	0.03123983343027
$r(4) = \tan^{-1}(-2^{-3})$	-0.12435499454676	$r(13) = \tan^{-1}(2^{-4})$	0.06241880999596
$r(5) = \tan^{-1}(-2^{-4})$	-0.06241880999596	$r(14) = \tan^{-1}(2^{-3})$	0.12435499454676
$r(6) = \tan^{-1}(-2^{-5})$	-0.03123983343027	$r(15) = \tan^{-1}(2^{-2})$	0.24497866312686
$r(7) = \tan^{-1}(-2^{-6})$	-0.01562372862048	$r(16) = \tan^{-1}(2^{-1})$	0.46364760900081
$r(8) = \tan^{-1}(-2^{-7})$	-0.00781234106010	$r(17) = \tan^{-1}(2^{-0})$	0.78539816339745
$r(9) = \tan^{-1}(0)$	0		

where

$N' \triangleq \sum_{i=0}^{N-1} \lvert\mu(i)\rvert$	*effective* iteration number.
$j, 0 \le j \le N'-1$	iteration index.
$s(j) \in \{0, 1, \ldots, N-1\}$	rotational sequence that determines the micro-rotation angle in the jth iteration.
$\alpha(j) \in \{-1, 0, 1\}$	directional sequence that controls the direction of the jth micro-rotation of $a(s(j))$.
$\tilde{\theta}(j)$	jth micro-rotation angle, defined as $\tilde{\theta}(j) \triangleq \tan^{-1}(\alpha(j) \cdot 2^{-s(j)})$.

As we can see from (12), the AR technique essentially tries to approximate θ with the combination of selected angle elements from a pre-defined elementary angle set (EAS). The EAS consists of all *possible* values of $\tilde{\theta}(j)$s, and the EAS $\mathcal{S}_1$ used in AR technique can be represented as

$$\mathcal{S}_1 = \left\{ \tan^{-1}\left(\alpha^\star \cdot 2^{-s^\star}\right) : \alpha^\star \in \{-1, 0, 1\}, \right. \\ \left. s^\star \in \{0, 1, \ldots, N-1\} \right\}. \quad (13)$$

The use of the subscript 1 will become apparent later in this section.

Here, we use an example to demonstrate the EAS $\mathcal{S}_1$. Each angle element (entry) in the set is denoted as $r(l)$. These angles are listed in Table IV in an ascending order based on their values. Note that EAS $\mathcal{S}_1$ covers not only all CORDIC angles in Table III, the *negative* elementary angle, $-a(i)$, is also treated as an *individual* angle of the EAS table. In other words, we can say that EAS used in conventional CORDIC algorithm is only a subset of EAS $\mathcal{S}_1$.

With the EAS $\mathcal{S}_1$ in hand, now we can easily link AR technique to the AQ process. By comparing (12) with the AQ approximation equation of (3), we find that AR technique indeed performs the angle quantization of target angle θ: The sub-angle θ_i now corresponds to $\tilde{\theta}(i) = \tan^{-1}(\alpha(i) \cdot 2^{-s(i)})$ and N_A is set to be N'.

Optimization Problem*:* We can also reconsider the optimization problem of AR technique from EAS $\mathcal{S}_1$ point of view. It can be restated as:

> Given θ, find the combination of elementary angles from EAS $\mathcal{S}_1$, such that the angle quantization error $|\xi_{m,AR}| \le a(N-1)$ and N' is minimized.

In summary, the AR technique intends to perform the angle quantization in a *more flexible way* than conventional CORDIC algorithm by relaxing the constraint of micro-rotation procedure. Hence, a faster and more precise rotation operation can be obtained.

B. MVR-CORDIC Algorithm [18]

Based on the AR technique, in the *Modified Vector Rotational CORDIC (MVR-CORDIC) algorithm* [18], two more modifications are proposed. These modifications also try to break the sequential rotation procedure of the conventional CORDIC algorithm so as to facilitate the operation of angle quantization. They are stated as follows.

1) *Repeat of Elementary Angles:*

Referring to (11), in the AR technique, each micro-rotation angle of $a(i) = \tan^{-1}(2^{-i})$ is allowed to be used *only once*. However, in the MVR-CORDIC algorithm, each micro-rotation of elementary angle can be performed repeatedly. The relaxed operation can result in more possible combinations of elementary angles, hence, smaller ξ_m can be expected. For example, we can rotation $\theta = 0.5261$ by performing micro-rotation of $a(2)$ once and $a(6)$ twice, i.e., $\theta \approx a(2) + a(6) + a(6)$, which, on the contrary, cannot be represented by (11) of AR technique.

2) *Confines of Total Micro-Rotation Number:*

From (12), we can see that the effective iteration number N' in the AR technique is not fixed. It will be varied with the target rotational angle θ. For certain cases, N' is large and very close to the upper bound of $N/2$ [17]. In synchronous circuit design, the overall speed performance of the CORDIC-based arithmetic operation will be limited by those special angles with large N'. Besides, the nonuniform feature of the iteration number is not suitable for modular design in VLSI implementation.

In the MVR-CORDIC algorithm, we confine the iteration number in the micro-rotation phase to R_m ($R_m \ll W$). The role of R_m is quite similar to the number of nonzero digits, N_D, used in CSD recoding scheme; it will dominate the precision performance and the complexity (see Table I).

1) Constellation of Reachable Angles: To see the effectiveness of the above modifications, we show the constellation of reachable angles of MVR-CORDIC algorithm in Fig. 6(b). The wordlength, W, is assigned to be 4, and the restricted iteration number, R_m, is 3. The reachable angles of conventional CORDIC for iteration number $N = 3$ ($N = R_m$) are also

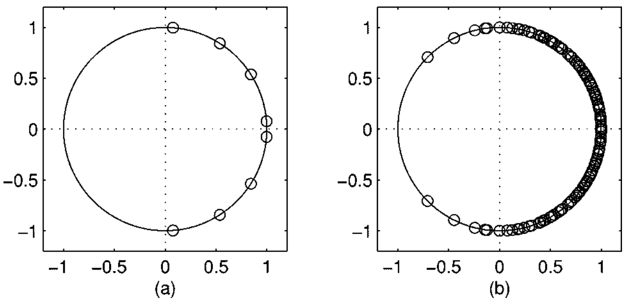

Fig. 6. Constellation of reachable angles of (a) conventional CORDIC with $N = R_m = 3$, (b) MVR-CORDIC algorithm with $W = 4$ and $R_m = 3$.

shown in Fig. 6(a) for comparison purpose. In fact, given W and R_m, the numbers of angles that can be represented by conventional CORDIC and MVR-CORDIC algorithm are in the order of 2^{R_m} and $(2W+1)^{R_m}$, respectively. Note that the comparison is made under the condition of *equal iteration number* in the micro-rotation phase.

As we can see in Fig. 6, the number of reachable angles of the MVR-CORDIC is much larger than the conventional CORDIC algorithm. This implies that given the same iteration number, the MVR-CORDIC algorithm will outperform the conventional CORDIC in terms of angle quantization error. Namely, given a target angle quantization error, the MVR-CORDIC rotation requires fewer iterations compared with conventional approach. Consequently, the speed performance of CORDIC-based arithmetic operations can be greatly improved.

2) AQ Process in MVR-CORDIC Algorithm:

Elementary Angle Set (EAS)*:* Next, we will explore the EAS in MVR-CORDIC algorithm. Putting all the aforementioned modifications together and ignoring the null operations, we can represent the angle quantization error of the MVR-CORDIC algorithm as

$$\xi_{m,MVR} \triangleq \theta - \left[\sum_{i=0}^{R_m-1} \alpha(i) a\left(s(i)\right) \right] \tag{14}$$

where

1) $s(i) \in \{0, 1, \ldots, W-1\}$ is the *rotational sequence* that determines the micro-rotation angle in the ith iteration.
2) $\alpha(i) \in \{-1, 0, 1\}$ is the *directional sequence* that controls the direction of the ith micro-rotation of $a(s(i))$.

As one can find that the sub-angle of $(\alpha(i)a(s(i)))$ in (14) is exactly the same as the definition of $\tilde{\theta}(j)$ in (12). Hence, the EAS formed by MVR-CORDIC algorithm is the same as AR technique, as shown by

$$\mathcal{S}_1 = \left\{ \tan^{-1}\left(\alpha^\star \cdot 2^{-s^\star}\right) : \alpha^\star \in \{-1, 0, 1\}, \quad s^\star \in \{0, 1, \ldots, N-1\} \right\}.$$

Based on the (14), it is obvious that MVR-CORDIC algorithm also performs the AQ process as well. The major difference is that

1) The total number of sub-angles N_A in Table I (i.e., the total iteration number in the micro-rotation phase) is now kept fixed to a pre-defined value of R_m ($N_A = R_m$)

2) The sub-angle θ_i corresponds to $\alpha(i)a(s(i))$ in MVR-CORDIC algorithm, i.e., $\theta_i = \alpha(i)a(s(i)) = \hat{\theta}(i)$.

TABLE V
SUMMARY OF THE MVR-CORDIC ALGORITHM

```
% Initialization
Given $x(0)$ and $y(0)$
% Micro-rotation phase
FOR $i = 0$ to $R_m - 1$
    $\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\alpha(i)2^{-s(i)} \\ \alpha(i)2^{-s(i)} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$
END
% Scaling phase
$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = P \begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix} = \left( \prod_{i=0}^{R_m-1} \sqrt{1+2^{-2s(i)}} \right)^{-1} \begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix}$
```

Optimization Problem*:* In the application of MVR-CORDIC algorithm, the optimization problem can be stated as follows.

> Given θ, find the sequences of $s(i) \in \{0, 1, \ldots, W-1\}$ and $\alpha(i) \in \{-1, 0, 1\}$, such that $|\xi_{m,MVR}|$ is minimized, subject to the constraint that the total iteration number is confined to R_m.

Similarly, the optimization problem can also be stated alternatively from EAS point of view as

> Given θ, find the combination of R_m elementary angles from EAS $\mathcal{S}_1$, such that the angle quantization error $|\xi_{m,MVR}|$ is minimized.

In Table V, we summarize the micro-rotation procedure and the scaling operation of the MVR-CORDIC algorithm. In summary, we may conclude that MVR-CORDIC algorithm is similar to the conventional CORDIC algorithm, except that the rotational constraint is greatly relaxed. It would be informative for readers to compare Table II with Table V in detail.

C. Extended EAS-Based CORDIC Algorithm [19]

In *Extended Elementary Angle Set (EEAS)*-based CORDIC algorithm [19], in addition to applying the relaxation on $\mu(i)$, we also relax the constraint of elementary angles by extending EAS $\mathcal{S}_1$. Then, we can have more choices (elementary angles) in approximating the target angle θ. It is expectable that the angle quantization error ξ_m can be reduced correspondingly.

Extended EAS (EEAS)*:* By observing (13), we can see that the EAS $\mathcal{S}_1$ are comprised of *arctangent of single signed-power-of-two (SPT) term,* i.e., $\tan^{-1}(\alpha^\star \cdot 2^{-s^\star})$. In the problem of SPT-based digital filter design, one effective way to increase the coefficient resolution (hence the filter performance) is to employ more SPT terms to represent the filter coefficients [12], [13]. Motivated by this, we can easily extend the set by representing the elementary angles as the *arctangent of the sum of two SPT terms* [19]. That is

$$\mathcal{S}_2 = \left\{ \tan^{-1}\left(\alpha_0^\star \cdot 2^{-s_0^\star} + \alpha_1^\star \cdot 2^{-s_1^\star}\right) : \quad \alpha_0^\star, \alpha_1^\star \in \{-1, 0, 1\}, s_0^\star, s_1^\star \in \{0, 1, \ldots, W-1\} \right\}. \tag{15}$$

We call it *Extended Elementary-Angle Set* $\mathcal{S}_2$ (EEAS $\mathcal{S}_2$). The subscript is used to denote the number of SPT terms.

In Table VI(b), we use an example ($W = 3$) to demonstrate the elementary angles in EEAS $\mathcal{S}_2$. The angles of EAS $\mathcal{S}_1$ are

TABLE VI

SUMMARY OF THE ELEMENTARY ANGLES OF (A) EAS $\mathcal{S}_1$ (B) EEAS $\mathcal{S}_2$ FOR THE CASE OF $W = 3$

Elementary Angle	Value in Radius	Elementary Angle	Value in Radius
$r(1) = \tan^{-1}(-2^{-0})$	-0.78539816339745	$r(5) = \tan^{-1}(2^{-2})$	0.24497866312686
$r(2) = \tan^{-1}(-2^{-1})$	-0.46364760900081	$r(6) = \tan^{-1}(2^{-1})$	0.46364760900081
$r(3) = \tan^{-1}(-2^{-2})$	-0.24497866312686	$r(7) = \tan^{-1}(2^{-0})$	0.78539816339745
$r(4) = \tan^{-1}(0)$	0		

(a)

Elementary Angle	Value in Radius	Elementary Angle	Value in Radius
$r(1) = \tan^{-1}(-2^{-0} - 2^{-0})$	-1.10714871779409	$r(9) = \tan^{-1}(2^{-2})$	0.24497866312686
$r(2) = \tan^{-1}(-2^{-0} - 2^{-1})$	-0.98279372324733	$r(10) = \tan^{-1}(2^{-1})$	0.46364760900081
$r(3) = \tan^{-1}(-2^{-0} - 2^{-2})$	-0.89605538457134	$r(11) = \tan^{-1}(2^{-1} + 2^{-2})$	0.64350110879328
$r(4) = \tan^{-1}(-2^{-0})$	-0.78539816339745	$r(12) = \tan^{-1}(2^{-0})$	0.78539816339745
$r(5) = \tan^{-1}(-2^{-1} - 2^{-2})$	-0.64350110879328	$r(13) = \tan^{-1}(2^{-0} + 2^{-2})$	0.89605538457134
$r(6) = \tan^{-1}(-2^{-1})$	-0.46364760900081	$r(14) = \tan^{-1}(2^{-0} + 2^{-1})$	0.98279372324733
$r(7) = \tan^{-1}(-2^{-2})$	-0.24497866312686	$r(15) = \tan^{-1}(2^{-0} + 2^{-0})$	1.10714871779409
$r(8) = \tan^{-1}(0)$	0		

(b)

also shown for comparison purpose [see Table VI(a)]. As the wordlength W increase, the size (number of elementary angles) of EEAS $\mathcal{S}_2$ increases exponentially. For $W = 8$, the number of elementary angles in EEAS $\mathcal{S}_2$ becomes 105, as opposed to 17 in EAS $\mathcal{S}_1$ of MVR-CORDIC algorithm (see Table IV).

Constellation of Reachable Angles in EEAS-Based CORDIC: Based on the EEAS $\mathcal{S}_2$ developed in (15), the sub-angle θ_i in Table I now can be represented as

$$\theta_i = \tan^{-1}\left(\alpha_0(j) \cdot 2^{-s_0(j)} + \alpha_1(j) \cdot 2^{-s_1(j)}\right) \tag{16}$$

and the number of sub-angles N_A is set to be R_m.

To see the effectiveness of the EEAS scheme, we show the constellation of all reachable elementary angles of $\mathcal{S}_1$ and $\mathcal{S}_2$ in Fig. 7(a) and (b), respectively. As we can see, the number of reachable angles of $\mathcal{S}_2$ is much larger than that of $\mathcal{S}_1$. This implies that EEAS $\mathcal{S}_2$ can yield better precision performance (smaller ξ_m) than $\mathcal{S}_1$ under a fixed number of micro-rotations of sub-angles (iterations).

Optimization Problem: With the derived EEAS $\mathcal{S}_2$, now the optimization problem of the EEAS-based CORDIC algorithm can be stated as

> Given θ and R_m, find the parameters of $\alpha_0(j)$, $\alpha_1(j)$, $s_0(j)$ and $s_1(j)$ (i.e., the combination of elementary angles from EEAS $\mathcal{S}_2$), such that the angle quantization error

$$|\xi_{m,EEAS}| \triangleq \left|\theta - \sum_{j=0}^{R_m-1} \tan^{-1}\left(\alpha_0(j) \cdot 2^{-s_0(j)} + \alpha_1(j) \cdot 2^{-s_1(j)}\right)\right| \tag{17}$$

> can be minimized.

In Table VII, we summarize the micro-rotation procedure and the scaling operation of the EEAS-based CORDIC scheme. Note that now four additions are required to complete each micro-rotation of elementary angle from $\mathcal{S}_2$, which is twice as many as in the conventional CORDIC approaches. Hence, the relaxation on the set of elementary angles is obtained at the expense of the doubled hardware/computational complexity. Fortunately, as will be shown in Section VI, the increased complexity can be compensated by the halved maximum iteration number.

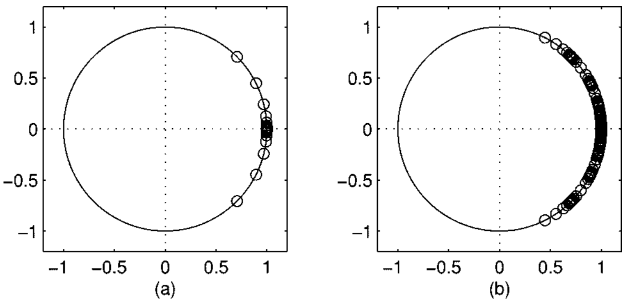

Fig. 7. Constellation of elementary angles of (a) set $\mathcal{S}_1$ (b) set $\mathcal{S}_2$, with wordlength $W = 8$.

D. Generalized EEAS Scheme

By following the similar idea of EEAS scheme, it is straightforward to insert more SPT terms in the representation of elementary angles. Hence, the size of EEAS can be increased. With more than two SPT terms, we call such an extension scheme *Generalized EEAS Scheme*. Specifically, the generalized EEAS with d SPT terms can be represented as

$$\begin{aligned}\mathcal{S}_d = \Big\{\tan^{-1}\left(\alpha_0^\star \cdot 2^{-s_0^\star} + \alpha_1^\star \cdot 2^{-s_1^\star} + \cdots + \alpha_{d-1}^\star \cdot 2^{-s_{d-1}^\star}\right) &: \\ \alpha_0^\star, \alpha_1^\star, \ldots, \alpha_{d-1}^\star \in \{-1, 0, 1\}&, \\ s_0^\star, s_1^\star, \ldots, s_{d-1}^\star \in \{0, 1, \ldots, W-1\}\Big\}&.\end{aligned} \tag{18}$$

As one can expect that the size of the EEAS increases exponentially as d increases. Consequently, with properly chosen design parameters, we can achieve higher precision performance in the AQ process.

TABLE VII
SUMMARY OF THE EEAS-BASED CORDIC SCHEME

```
% Initialization
Given initial vector of $[x(0), y(0)]^T$
% Micro-rotation phase
FOR $j = 0$ to $R_m - 1$
    $\begin{bmatrix} x(j+1) \\ y(j+1) \end{bmatrix} = \begin{bmatrix} 1 & -\alpha_0(j)\cdot 2^{-s_0(j)} - \alpha_1(j)\cdot 2^{-s_1(j)} \\ \alpha_0(j)\cdot 2^{-s_0(j)} + \alpha_1(j)\cdot 2^{-s_1(j)} & 1 \end{bmatrix} \begin{bmatrix} x(j) \\ y(j) \end{bmatrix}$
END
% Scaling phase
$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = P \begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix} = \frac{1}{\prod_{j=0}^{R_s-1} \sqrt{1 + [\alpha_0(j)\cdot 2^{-s_0(j)} + \alpha_1(j)\cdot 2^{-s_1(j)}]^2}} \begin{bmatrix} x(R_m) \\ y(R_m) \end{bmatrix}$
```

TABLE VIII
COMPARISONS OF MEMBERS IN THE VECTOR ROTATIONAL CORDIC FAMILY

Vector Rotation Algorithms	Selection of Rotation Sequence	Elementary Angle Set	Micro-rotations	Angle Quantization	
				θ_i	N_A
Conventional CORDIC Algorithm	$\mu \in \{-1,1\}$	EAS S	Complete	$\theta_i = \mu(i)a(i)$	W Fixed
Angle Recoding Technique	$\mu \in \{-1,0,1\}$	EAS S_1	Selective	$\theta_i = \tan^{-1}\left(\alpha(i)\cdot 2^{-s(i)}\right)$	N' Variable
MVR-CORDIC Algorithm	$\alpha \in \{-1,0,1\}$	EAS S_1	Selective	$\theta_i = \tan^{-1}\left(\alpha(i)\cdot 2^{-s(i)}\right)$	R_m Fixed
EEAS Scheme	$\alpha_0, \alpha_1 \in \{-1,0,1\}$	EEAS S_2	Selective	$\theta_i = \tan^{-1}\left(\alpha_0(i)\cdot 2^{-s_0(i)} + \alpha_1(i)\cdot 2^{-s_1(i)}\right)$	R_m Fixed
Generalized EEAS Scheme	$\alpha_0, \alpha_1, \cdots, \alpha_{d-1} \in \{-1,0,1\}$	EEAS S_d $d \geq 3$	Selective	$\theta_i = \tan^{-1}\left(\alpha_0(i)\cdot 2^{-s_0(i)} \cdots + \alpha_{d-1}(i)\cdot 2^{-s_{d-1}(i)}\right)$	R_m Fixed

E. Family of Vector Rotational CORDIC Algorithm

So far, we have linked the AQ process with several existing vector rotation approaches, including CORDIC algorithm, Angle Recoding technique, MVR-CORDIC algorithm, EEAS scheme, and Generalized EEAS scheme. All algorithms intend to realize the AQ process with various EAS and suitable combinations of sub-angles. That is, they try to decompose the target rotational angle into several easy-to-implement sub-angles, while minimizing the angle quantization error ξ_m to obtain the best precision performance.

Based on our discussion, now we can link all these rotation algorithms together under a unified design framework, from the AQ point of view. They form a family of vector rotational CORDIC algorithm, called *Vector Rotational CORDIC Family*. They all conform to the AQ process, but each rotational algorithm uses different AQ setting as summarized in Table VIII.

Note that EEAS scheme covers MVR-CORDIC algorithm and AR technique due to the fact that MVR-CORDIC and AR employ EAS $\mathcal{S}_1$ as a searching space that is a subset of EEAS $\mathcal{S}_2$. Moreover, MVR-CORDIC algorithm can also be treated as a subset of AR technique due to the fact that we impose one constraint on the total iteration number.

Fig. 8 illustrates the relationships among members of vector rotational CORDIC family.

IV. SEARCHING ALGORITHMS FOR THE VECTOR ROTATIONAL CORDIC FAMILY

In the applications of vector rotational CORDIC family, optimization of the *angle quantization error* are always encountered. The problem is conceptually analogous to finding the closest codeword for a coefficient, h_i, under given number of nonzero digits, N_D, in the CSD application. In this section, we address four searching algorithms. They are *Greedy algorithm*, *Exhaustive Searching algorithm*, *Semi-greedy algorithm*, and *Trellis-based Searching algorithm*, respectively. The target optimization problem is the EEAS-based CORDIC algorithm derived in Section III-C [see (17)]. We choose it since it covers most other rotational schemes, as illustrated in Fig. 8.

A. Greedy Algorithm

In [17], *Greedy algorithm* has been proposed to solve the AR optimization problem. In running the Greedy algorithm, we approach the target rotation angle, θ, step by step. That is, in each

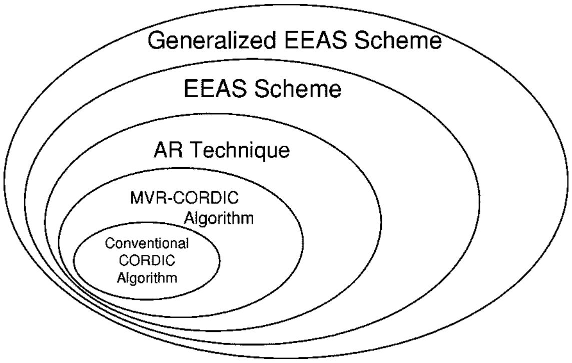

Fig. 8. Set diagram of vector rotational CORDIC family.

searching step, without looking ahead, decisions are made on θ_i [i.e., $\alpha_0(i)$, $\alpha_1(i)$, $s_0(i)$, and $s_1(i)$] so that the accumulated θ_is is the best approximation of the target angle θ. Specifically, θ_i is determined such that the error function of $J(i) = |\rho(i) - \theta_i|$ is minimized, where $\rho(i)$ is the residue angle at ith step, defined as

$$\rho(i) = \theta - \sum_{m=0}^{i-1} \theta_m. \tag{19}$$

The searching algorithm is terminated if no further improvement can be found, i.e., $J(i) \geq J(i-1)$; or $\theta_{R_m - 1}$ is determined at the end of the searching process. Most of the time, Greedy algorithm gives us local optimal solution.

B. Exhaustive Searching Algorithm

The idea of Exhaustive searching algorithm is to search for the entire solution space, i.e., all the possible combinations of $\sum_{i=0}^{R_m-1} \theta_i$. Decisions on θ_i, for $0 \leq i \leq R_m - 1$, are made such that the error function

$$J = \left| \theta - \sum_{i=0}^{R_m-1} \theta_i \right| \tag{20}$$

is minimized. Obviously, the Exhaustive searching algorithm produces global optimal solution, but it is too time-consuming in finding the design parameters.

C. Semi-Greedy Algorithm [18]

Basically, we can treat the Semi-greedy algorithm as a combination of Greedy and Exhaustive searching algorithm. The whole searching space of θ_i for $0 \leq i \leq R_m - 1$ is divided into several segments, and each segment contains D iterations. We call such a segment as a *block*, and D denotes the *block length*. The concept of segmentation scheme is illustrated in Fig. 9. In the Semi-greedy algorithm, the exhaustive search is performed within each isolated block, and the connection between each consecutive blocks is determined in the greedy manner. Specifically, in the ith block (corresponds to ith *step* in performing the searching algorithm), the decisions of θ_k for $iD \leq k \leq (i+1)D - 1$ are made to minimize the error function

$$J(i) = \left| \rho(i) - \sum_{k=iD}^{(i+1)D-1} \theta_k \right| \tag{21}$$

where

$$\rho(i) = \theta - \sum_{m=0}^{i-1} \left[\sum_{k=mD}^{(m+1)D-1} \theta_k \right] \tag{22}$$

is the residue angle at the ith step. For simplicity of representation, we assume R_m is a multiple of D in (21) and (22).

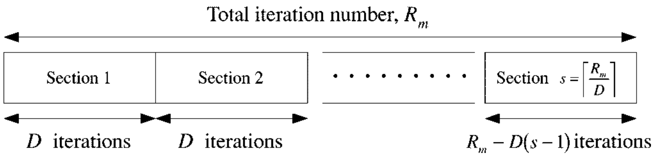
Total iteration number, R_m
Section 1
Section 2
. . . .
Section $s = \left\lceil \frac{R_m}{D} \right\rceil$
D iterations
D iterations
$R_m - D(s-1)$ iterations

Fig. 9. Segmentation of the searching space with block length, D, in the Semi-greedy algorithm.

D. Trellis-Based Searching (TBS) Algorithm

The trellis-based searching (TBS) algorithm provides an alternative way to solve the complicated optimization problem [20]. Here, the TBS algorithm is applied to solve the optimization problem encountered in the EEAS-based CORDIC algorithm.

Step 1) Initialization

First of all, let $Z(\mathcal{S}_2)$ denote the number of the elementary angles in the extended set $\mathcal{S}_2$, and each distinct elementary angle in the set is expressed as $r(k)$, for $1 \leq k \leq Z(\mathcal{S}_2)$, i.e., $\mathcal{S}_2 = \{r(1), r(2), \ldots, r(Z(\mathcal{S}_2))\}$. In the TBS algorithm, there are $Z(\mathcal{S}_2)$ states in each step. For kth state ($1 \leq k \leq Z(\mathcal{S}_2)$) of ith search step, we use the *Cumulative Angle*, $\phi(i, k)$, to denote the *best approximation* of angle θ in the kth state up to the ith step. The TBS algorithm is performed column-wise from left to right. Initially, we start the TBS algorithm by setting all $\phi(1, k)$ as the corresponding elementary angles, i.e., $\phi(1, k) = r(k)$ for all k.

Step 2) Accumulation

A path in the trellis, which leaves the kth state at ith step and enters the k'th state at $(i+1)$th step, corresponds to an operation of adding $\phi(i, k)$ by $r(k')$. Then, the appended angle of $\phi(i, k) + r(k')$ becomes the candidate for $\phi(i+1, k')$. Moreover, from a given state at step i^*, the paths can diverge to all the states at the next search step $(i^* + 1)$. Namely, there are $Z(\mathcal{S}_2)$ paths, carrying the corresponding appended angles of $\phi(i^*, k) + r(k')$ for all k, enters the k'th state at $(i^* + 1)$th step. Then, those appended angles form the candidate set for the cumulative angle of $\phi(i^* + 1, k')$.

Step 3) Comparison and Selection

Conceptually, the whole process is similar to the trellis decoding of convolutional code [21]: The TBS algorithm involves calculating and minimizing the difference between the target angle θ and $\phi(i,\, k)$ for all k at each search step i. To be specific, $\phi(i+1,\, k)$ is determined such that

$$|\phi(i+1,\, k) - \theta| = \min_{1 \le k^\star \le Z(\mathcal{S}_2)} |\phi(i,\, k^\star) + r(k) - \theta|\,. \quad (23)$$

Then, the selected path is denoted as the *surviving path*. Note that we have to calculate all the cumulative angles $\phi(i,\, k)$ for all k (thus their corresponding surviving paths) before moving to the $(i+1)$th step. Continuing in this manner, we can successively advance deeper into the trellis (set $i = i+1$), until the maximum iteration number is reached ($i = R_m$).

Step 4) Determination of the Global Result and Trace Back

After calculating all the cumulative angles at the last search step, the next procedure for the TBS algorithm is to determine the *global result*, θ_{TBS}. Similar to the determination of surviving path, we decide θ_{TBS} as follows:

$$|\theta_{TBS} - \theta| = \min_{1 \le k^\star \le Z(\mathcal{S}_2)} |\phi(R_m,\, k^\star) - \theta|\,. \quad (24)$$

Next, we can determine all the micro-rotations by tracing from the state, whose corresponding $\phi(R_m,\, k)$ is best approximation of θ, along its surviving path backward.

V. SCALING OPERATION FOR THE VECTOR ROTATIONAL CORDIC FAMILY

In this section, we consider the indispensable phase in the practical implementation of vector rotational CORDIC family—*the scaling phase*. As mentioned in Section II, each micro-rotation (rotation of sub-angles) enlarges the norm of the vector by a factor κ_i. The factor κ_i depends on which algorithm in the vector rotational CORDIC family is employed. For example, $\kappa_i = \sqrt{1 + 2^{-2s(i)}}$ in the AR technique and MVR-CORDIC algorithm; and

$$\kappa_i = \sqrt{1 + \left(\alpha_0(i) \cdot 2^{-2s_0(i)} + \alpha_1(i) \cdot 2^{-2s_1(i)}\right)^2}$$

in EEAS-based CORDIC scheme. After applying a sequence of R_m micro-rotations of sub-angles, the norm of the rotated vector will be enlarged by a factor of $\prod_{i=0}^{R_m-1} \kappa_i$. Consequently, to preserve the norm of $[x(0)\ y(0)]^T$, we have to scale the rotated vector of $[x(R_m)\ y(R_m)]^T$ by the *scaling factor* of

$$P = \left(\prod_{i=0}^{R_m-1} \kappa_i\right)^{-1}. \quad (25)$$

A. Scaling Operations

In practical implementation, several scaling operations has been proposed to save the hardware complexity; they are *Type-I and Type-II scaling operations* [15], [16], [14], and the *Extended Type-II (ET-II) scaling operation* [19]. These scaling operations are performed by quantizing the floating-based scaling factor, P, in the following fixed-point forms:

$$\text{Type-I:} \quad \hat{P} = \sum_{m=0}^{R_s-1} k(m) \cdot 2^{-q(m)} \quad (26)$$

$$\text{Type-II:} \quad \hat{P} = \prod_{m=0}^{R_s-1} \left[1 + k(m) \cdot 2^{-q(m)}\right] \quad (27)$$

$$\text{ET-II:} \quad \hat{P} = \prod_{m=0}^{R_s-1} \Big[1 + k_0(m) \cdot 2^{-q_0(m)} + k_1(m) \cdot 2^{-q_1(m)}\Big]. \quad (28)$$

Here, $\hat{P}$ denotes the *quantized* value of P, $k(m), k_0(m),\ k_1(m) \in \{1, -1\}$, and $q(m), q_0(m), q_1(m) \in \{0, 1, \ldots, W-1\}$. R_s is the counterpart of R_m in the micro-rotation phase, determining the number of iterations in the scaling phase. The corresponding iteration procedures for those scaling operations are listed as follows:

$$\text{Type-I:} \begin{cases} \tilde{x}(m+1) = \tilde{x}(m) + k(m) \cdot 2^{-q(m)} \cdot x(R_m) \\ \tilde{y}(m+1) = \tilde{y}(m) + k(m) \cdot 2^{-q(m)} \cdot y(R_m) \end{cases} \quad (29)$$

for $m = 0, 1, \ldots, R_s - 1$, with $\tilde{x}(0) = 0$ and $\tilde{y}(0) = 0$

$$\text{Type-II:} \begin{cases} \tilde{x}(m+1) = \tilde{x}(m) + k(m) \cdot 2^{-q(m)} \cdot \tilde{x}(m) \\ \tilde{y}(m+1) = \tilde{y}(m) + k(m) \cdot 2^{-q(m)} \cdot \tilde{y}(m) \end{cases} \quad (30)$$

for $m = 0, 1, \ldots, R_s - 1$, with $\tilde{x}(0) = x(R_m)$ and $\tilde{y}(0) = x(R_m)$

$$\text{ET-II:} \begin{cases} \tilde{x}(m+1) = \tilde{x}(m) \\ \qquad + \left[k_0(m) \cdot 2^{-q_0(m)} + k_1(m) \cdot 2^{-q_1(m)}\right] \cdot \tilde{x}(m) \\ \tilde{y}(m+1) = \tilde{y}(m) \\ \qquad + \left[k_0(m) \cdot 2^{-q_0(m)} + k_1(m) \cdot 2^{-q_1(m)}\right] \cdot \tilde{y}(m) \end{cases} \quad (31)$$

for $m = 0, 1, \ldots, R_s - 1$, with $\tilde{x}(0) = x(R_m)$ and $\tilde{y}(0) = y(R_m)$. Here, $\tilde{x}(m)$ and $\tilde{y}(m)$ are used in the scaling phase to distinguish them from the ones in micro-rotation phase.

After R_s iterations, the scaled rotational vector becomes $[x_f\ y_f]^T = [\tilde{x}(R_s)\ \tilde{y}(R_s)]^T$. By doing so, we can *approximate* the scaling factor of P by using only $2R_s$ shift-and-add operations for Types-I and -II scaling operation, and $4R_s$ shift-and-add operations for the ET-II scaling operations. Moreover, it is advantageous to employ Types-I or -II scaling operation when AR technique and MVR-CORDIC algorithm are employed as the vector rotation kernel. By doing this, scaling operation can share the same VLSI circuits with the micro-rotation module, which can save the significant hardware overhead of scaling multipliers. Similarly, for the case the EEAS-based CORDIC scheme, ET-II scaling operation is preferred due to the hardware sharing feature.

B. *Scaling Approximation Error and SQNR Performance*

With the approximated $\hat{P}$, the scaling operations will introduce some quantization noise, and it will increases as R_s decreases. Similar to the micro-rotation phase, we introduce another performance index, ξ_s, to describe the amount of error introduced by the approximation process of (26)–(28). The *scaling approximation error*, ξ_s, is defined as

$$\xi_s \triangleq \left|1 - \frac{\hat{P}}{P}\right|. \tag{32}$$

Now we have introduced two indices of ξ_m and ξ_s to describe the precision performance of vector rotational CORDIC family in two separate phases. Nevertheless, for most DSP applications, the precision performance is usually represented in terms of *signal to quantization-noise ratio* (SQNR), defined as

$$\mathrm{SQNR\ (dB)} \triangleq 10\log_{10}\left[\frac{\text{Signal Variance}}{\text{Quantization Noise Variance}}\right]. \tag{33}$$

The usage of SQNR can give designers a more straightforward view about the signal quality in practical fixed-point implementations. It has been proven that we can relate the parameters of ξ_m and ξ_s to SQNR performance as [18]

$$\mathrm{SQNR\ (dB)} = 10\log_{10}\left(\frac{1}{\xi_m^2 + \xi_s^2}\right). \tag{34}$$

The close-form equation links both error indices in a very concise way.

VI. PERFORMANCE COMPARISON AND DESIGN EXAMPLE

In this section, we compare the precision performance for the aforementioned vector rotational algorithms and searching algorithms.

A. *EEAS Scheme versus MVR-CORDIC Algorithm*

Essentially, the comparison between EEAS Scheme and MVR-CORDIC algorithm is the comparison between EEAS $\mathcal{S}_2$ versus EAS $\mathcal{S}_1$. In Section VII, we will show that it takes four full adders (FAs) to perform each rotation of sub-angle in the EEAS scheme, as opposed to two FAs in the MVR-CORDIC algorithm. Hence, to make fair comparison, the performance of these two approaches must be evaluated under the condition that *the maximum number of FAs are the same*. Denote the maximum iteration number of EEAS scheme and MVR-CORDIC algorithm as $R_{m,\,\mathrm{EEAS}}$ and $R_{m,\,\mathrm{MVR}}$, respectively. They need to satisfy the constraint of

$$R_{m,\,\mathrm{EEAS}} = R_{m,\,\mathrm{MVR}}/2. \tag{35}$$

In the first simulation, 4097 uniformly spaced rotation angles in the region from 0 to $\pi/2$, i.e., $\theta = 0, (1\cdot\pi/8192), (2\cdot\pi/8192), \ldots, (4096\cdot\pi/8192)$, are performed. The Greedy algorithm is adopted to solve the optimization problem, and the wordlength $W = 16$. The simulation results are shown in Fig. 10, where $\xi_{m,\,\mathrm{MVR}}$ and $\xi_{m,\,\mathrm{EEAS}}$ denote the ensemble-averaged angle quantization error of the MVR-CORDIC algorithm and the EEAS scheme, respectively.

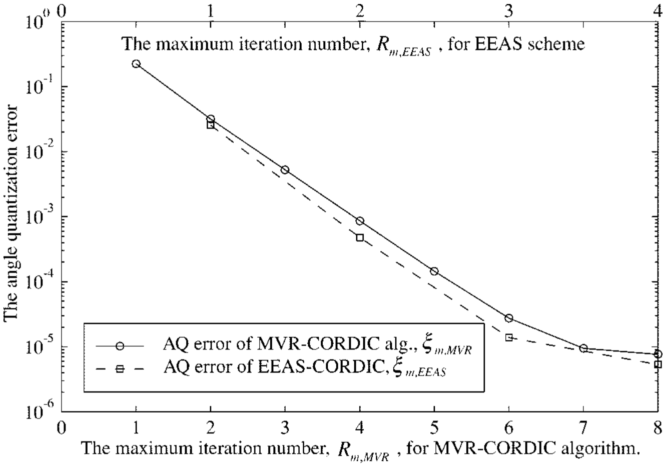

Fig. 10. Performance comparison between EEAS scheme and MVR-CORDIC algorithm.

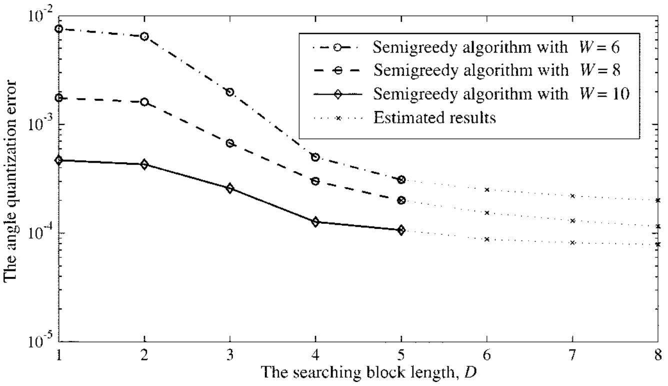

Fig. 11. Error performance versus searching block length, D, in Semi-greedy algorithm ($W = 6$, $W = 8$, $W = 10$ and $R_m = 3$).

From Fig. 10, we can make the following observations:

1) Increasing the maximum iteration number ($R_{m,\,MVR}$ or $R_{m,\,\mathrm{EEAS}}$) has the effect of improving error performance. This is consistent with the conventional CORDIC algorithm.

2) The results show that the EEAS scheme outperforms the MVR-CORDIC algorithm when the comparison criteria, (35), is satisfied. This can be explained that the EAS $\mathcal{S}_1$ in (13) is only a subset of $\mathcal{S}_2$ in (15). Any possible combination of elementary angles from $\mathcal{S}_1$ can also be constructed by using set $\mathcal{S}_2$.

B. *Error Performance versus Searching Block Length, D*

Fig. 11 depicts the angle quantization error versus the searching block length D in semi-greedy algorithm. The comparison is intended to highlight the precision performance for various block length. From the results shown in Fig. 11, we can make the following observations:

1) We can obtain better performance by increasing the searching block length of semi-Greedy algorithm. The results confirm our argument in Section IV-C that for larger D, the

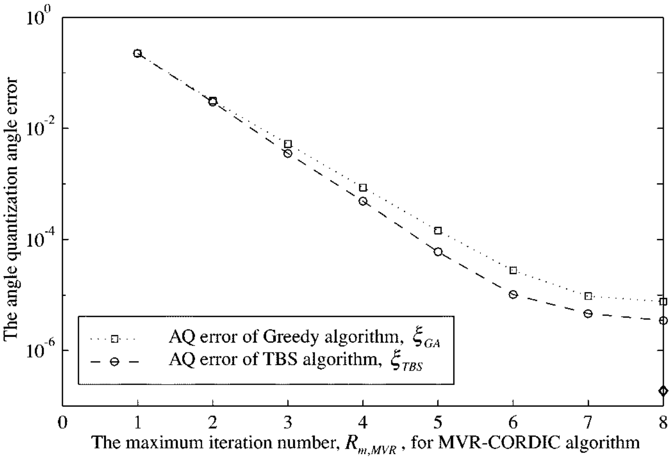

Fig. 12. Performance comparison between Greedy algorithm and TBS algorithm.

Semi-greedy behaves like Exhaustive searching algorithm. On the other hand, when D is small, the essence of Greedy algorithm will arise due to the confined searching space.

2) The saturation phenomenon suggests that we can use semi-Greedy algorithm with moderate value of D (say $D = 5$ in this case) to obtain a near-optimal error performance without going through exhaustive search.

Meanwhile, the saving in computational complexity is significant.

C. Trellis-Based Searching (TBS) Algorithm versus Greedy Algorithm

In this simulation, both Greedy algorithm and TBS algorithms are applied to solve the optimization problem of MVR-CORDIC algorithm. Let ξ_{GA} and ξ_{TBS} denote the angle quantization error generated by the Greedy and TBS algorithms, respectively. The simulation results are shown in Fig. 12.

Based on Fig. 12, we can see the following.

1) For any $R_{m,\,\mathrm{MVR}}$, the error performance of the proposed TBS algorithm is superior to the Greedy algorithm. Actually, we have proved in [18] that the precision performance of TBS algorithm always outperforms that of Greedy algorithm. For the case of $R_{m,\,\mathrm{MVR}} = 1$, it can be easily shown that the operations of TBS algorithm is identical to the Greedy algorithm. Hence, the error performance is the same.

2) The improvement of the error performance of TBS algorithm become more significant as maximum iteration number, $R_{m,\,\mathrm{MVR}}$, increases. The reason is that as $R_{m,\,\mathrm{MVR}}$ increases, more possible combinations of the elementary angles can be found by the TBS algorithm than the Greedy algorithm.

D. Design Example

In the design example, we consider the rotation angle of $\theta = 13\pi/32$. All algorithms in vector rotational CORDIC family derived in Section III are applied to perform the rotation of θ. Meanwhile, searching algorithms in Section IV are conducted to solve the optimization problems. In particular, we consider the following the combinations:

- Conventional CORDIC algorithm [14]–[16]
- AR technique + Greedy algorithm [17]
- MVR-CORDIC algorithm + Greedy algorithm with $D = 2$
- MVR-CORDIC algorithm + Semi-greedy algorithm with $D = 2$
- MVR-CORDIC algorithm + TBS algorithm
- EEAS scheme + Greedy algorithm
- EEAS scheme + TBS algorithm.

In Table IX, we summarize the results for these aforementioned combinations, including rotational parameters (μ, α, and s) and the angle approximation error ξ_m. Based on the results, we may make the following the observations:

1) In AR technique, it takes 6 iterations to complete the rotation, as oppose to 16 iterations in conventional CORDIC algorithm. Moreover, the angle quantization error of AR technique is smaller than that of CORDIC algorithm.

2) By comparing combinations 3 and 5 as well as the combinations 6 and 7, we find the TBS algorithm can generate superior precision performance than that of Greedy algorithm. By comparing combinations 3 and 4, Semi-greedy algorithm is also found to be better than Greedy algorithm.

3) By comparing combinations 3 and 6 as well as the combinations 5 and 7, we find that we can obtain better precision performance using EEAS $\mathcal{S}_2$ than EAS $\mathcal{S}_1$. Note that the comparison is made under the criteria that $R_{m,\,MVR} = 2{\cdot}R_{m,\,\mathrm{EEAS}}$.

4) The combination 8 has superior performance than combination 2. The comparison is made under the condition of *equal number of FAs*. It takes 6 iterations to perform the rotation of θ in combination 2, thus 12 FA's are required. The number is equal to that of EEAS scheme with $R_m = 3$.

As to the scaling phase, we take the combination of EEAS scheme with $R_m = 2$ and TBS algorithm (combination 7) as the example. By substituting the rotation parameters of $\alpha_0(i)$, $\alpha_1(i)$, $s_0(i)$, and $s_1(i)$ into (25), we can calculate the scaling factor as $P = 0.7002$. As mentioned earlier that when the EEAS-based CORDIC algorithm is employed in the micro-rotation phase, it is advantageous to use the ET-II scaling operation in the scaling phase. That is, we try to represent P in the fixed-point form of (28). Actually, the parameters of $k_0(m)$, $k_1(m)$, $q_0(m)$ and $q_1(m)$ can be determined with the help of TBS algorithm due the similarity between ET-II scaling operation and EEAS scheme [19]. In this example, assuming that the iteration number in scaling phase is $R_s = 2$, we derive the parameters in ET-II scaling operation as $\overline{k}_0 = [-1, 1]$, $\overline{q}_0 = [4, 10]$, $\overline{k}_1 = [-1, -1]$, and $\overline{q}_1 = [2, 8]$. Then, the scaling approximation error of $\xi_s = 7.241 * 10^{-6}$. Next, we can substitute the ξ_m and ξ_s into (34), and the SQNR value of the scaled EEAS-based CORDIC algorithm is about 95 dB.

VII. VLSI IMPLEMENTATION OF VECTOR ROTATIONAL CORDIC FAMILY

In this section, we will illustrate the VLSI structure (also known as CORDIC processor) for the proposed vector rotational CORDIC family. It is derived based on the conventional CORDIC VLSI architecture in [14].

TABLE IX
DESIGN EXAMPLE OF ROTATION ANGLE $\theta = 13\pi/32$, WHERE THE WORDLENGTH $W = 16$

Rotaion Angle $\theta = 13\pi/64$	**Searching Algorithm**	**Combination Type**	**Rotational Sequence** μ **and** α **and Subangle Index** s	**Angle Approx. Error** ξ_m
Conventional CORDIC Algorithm	-------------------	1	$\bar{\mu} = [1\ -1\ 1\ 1\ -1\ 1\ -1\ -1\ 1\ -1\ -1\ 1\ 1\ -1\ 1\ 1]$	$1.1608 * 10^{-5}$
Angle Recoding Technqiue	Greedy Algorithm	2	$\bar{\mu} = [1\ 0\ 0\ -1\ 0\ 0\ -1\ -1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1]$	$1.0110 * 10^{-5}$
MVR-CORDIC Algorithm with $R_m = 4$	Greedy Algorithm	3	$\bar{\alpha} = [1\ -1\ -1\ -1]$ $\bar{s} = [0\ 3\ 6\ 7]$	$5.2891 * 10^{-4}$
	Semi-greedy Algorithm ($D = 2$)	4	$\bar{\alpha} = [1\ -1\ -1\ 1]$ $\bar{s} = [0\ 3\ 5\ 7]$	$5.2033 * 10^{-4}$
	TBS Algorithm	5	$\bar{\alpha} = [1\ 1\ -1\ -1]$ $\bar{s} = [1\ 2\ 4\ 7]$	$2.5911 * 10^{-4}$
EEAS Scheme with $R_m = 2$	Greedy Algorithm	6	$\bar{\alpha}_0 = [1\ -1]$ $\bar{\alpha}_1 = [-1\ -1]$ $\bar{s}_0 = [0\ 2]$ $\bar{s}_1 = [8\ 10]$	$4.8233 * 10^{-4}$
	TBS Algorithm	7	$\bar{\alpha}_0 = [1\ 1]$ $\bar{\alpha}_1 = [-1\ -1]$ $\bar{s}_0 = [0\ 6]$ $\bar{s}_1 = [3\ 5]$	$1.7196 * 10^{-5}$
EEAS Scheme with $R_m = 3$	TBS Algorithm	8	$\bar{\alpha}_0 = [1\ -1\ -1]$ $\bar{\alpha}_1 = [1\ -1\ 1]$ $\bar{s}_0 = [0\ 3\ 7]$ $\bar{s}_1 = [15\ 6\ 12]$	$3.2503 * 10^{-7}$

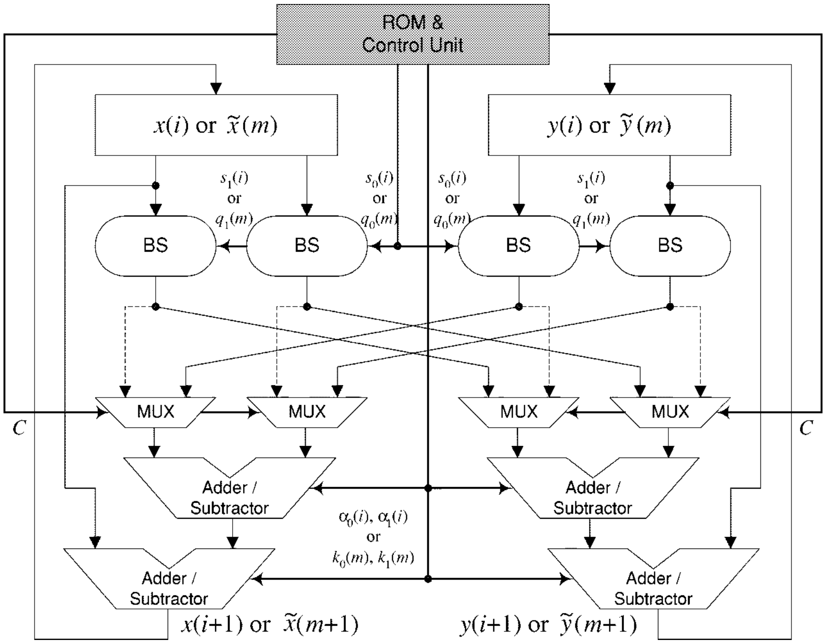

Fig. 13. The iterative architecture of the vector rotational CORDIC algorithm.

A. Iterative Architecture

In Fig. 13, we develop the iterative architecture for EEAS-based CORDIC algorithm. It consists of 4 barrel shifters (BS), 4 multiplexers (MUX) and 4 adders/substrators. Here, we employ the EEAS scheme as the architectural design basis. This reason is that we can easily modify the iterative VLSI architecture of Fig. 13 to AR and MVR-CORDIC-based architecture by removing one BS, one MUX and two adders/substrators. Of course, control signals as well as signal paths must be modified correspondingly. On the other hand, we can insert more BS's, MUX's and adders/substrators to realize the Generalized EEAS scheme.

As shown in Fig. 13, two separate phases are performed to complete single vector rotation, i.e., the micro-rotational phase (marked by solid line) and the scaling phase (marked by dash line). In each phase, three kinds of control signal are used to control the operations.

- $s_0(i)$ and $s_1(i)$ in micro-rotation phase as well as $q_0(m)$, $q_1(m)$ in scaling phase: they control the number of bits to be shifted by barrel shifters.

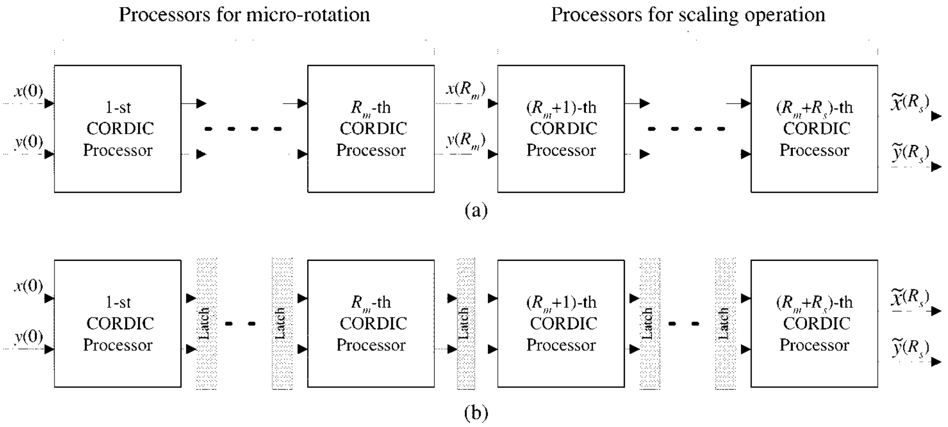
Processors for micro-rotation
Processors for scaling operation
x(0)
y(0)
1-st CORDIC Processor
R_m-th CORDIC Processor
$x(R_m)$
$y(R_m)$
(R_m+1)-th CORDIC Processor
(R_m+R_s)-th CORDIC Processor
$\tilde{x}(R_s)$
$\tilde{y}(R_s)$
(a)
Latch
(b)

Fig. 14. (a) Parallel and (b) pipelined architecture of the vector rotational CORDIC algorithm.

- $\alpha_0(i)$ and $\alpha_1(i)$ in micro-rotation phase as well as $k_0(m)$, $k_1(m)$ in scaling phase: they determine the operations of adders/subtracters.
- Control signal, C: it determines the data path by controlling the multiplexer; governing the phase switching of the iterative CORDIC architecture.

All the control signals can be generated by searching algorithms in advance, and are stored in ROM.

B. Parallel and Pipelined Architecture

By unfolding the iterative implementation of Fig. 13, we can obtain the parallel structure as depicted in Fig. 14(a). The structure is composed of $(R_m + R_s)$ EEAS-based CORDIC processors connected in cascade form, in which the R_m leading processors perform the micro-rotations and the following R_s processors execute the scaling operations. Each basic processor performs one iteration as specified in Fig. 13. Moreover, for the case that the parallel structure is dedicated to perform a given rotation angle, the operation of each processor can be kept fixed. We can thus save the hardware complexity easily by replacing all the control circuits, barrel shifters, and multiplexers with only wire routing.

To achieve a higher data throughput rate, we can further insert pipeline stages (latches) between successive processors of parallel structure, which results in the pipelined architecture in Fig. 14(b). Due to the reduced critical path, the pipelined structure is very suitable for real-time applications at high data bandwidth.

VIII. CONCLUSION

In this paper, we introduce a new design index, called Angle Quantization. Following the new index, we propose a unified design framework for several existing vector rotational algorithms. With the versatile feature of the design framework, we can identify more design parameters. Hence, designers can explore a bigger design space in deriving low-cost/high-performance rotational circuits. As illustrated in [6], most popular DSP algorithms can be realized via rotational circuits. The new framework proposed in this paper can be employed to design the processing kernel of the DSP engine in [6].

REFERENCES

[1] Y. H. Hu and Z. Wu, "An efficient CORDIC array structure for the implementation of discrete cosine transform," *IEEE Trans. Signal Processing*, vol. 43, pp. 331–336, Jan. 1995.
[2] J. H. Hsiao, L. G. Ghen, T. D. Chiueh, and C. T. Chen, "High throughput CORDIC-based systolic array design for the descrete cosine transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 218–225, Jan. 1995.
[3] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Trans. Computers*, vol. 23, pp. 993–1001, Oct. 1974.
[4] ——, "Very fast Fourier transform algorithms for hardware implementation," *IEEE Trans. Computers*, vol. 28, pp. 333–341, May 1979.
[5] P. P. Vaidyanathan, "A unified approach to orthogonal digital filters and wave digital filters based on the LBR two-pair extraction," *IEEE Trans. Circuits Syst.*, pp. 673–686, July 1985.
[6] A. Y. Wu, K. J. R. Liu, and A. Raghupathy, "System architecture of an adaptive reconfigurable DSP computing engine," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 54–73, Feb. 1998.
[7] J. Hormigo, J. Villalba, and E. Zapata, "Interval sine and cosine functions computation based on variable-precision CORDIC algorithm," in *Proc. 14th IEEE Symp. on Computer Arithmetic, 1999*, 1999, pp. 186–193.
[8] A. Madisetti, A. Kwentus, and A. J. Willson, "A sine/cosine direct digital frequency synthesizer using an angle rotation algorithm," in *Dig. Tech. Papers. 41st ISSCC IEEE Int. Solid-State Circuits Conf.*, 1995, pp. 262–263.
[9] J. Vankka, M. Kosunen, J. Hubach, and K. Halonen, "A CORDIC-based multicarrier QAM modulator," in *Global Telecommunications Conf.*, 1999, pp. 173–177.
[10] J. Vankka, M. Kosunen, I. Sanchis, and K. Halonen, "A multicarrier QAM modulator," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 1–10, Jan. 2000.
[11] K. Hwang, *Computer Arithmetic: Principles, Architecture and Design*. New York: Wiley, 1979.
[12] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with power-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044–1047, July 1989.
[13] Y. C. Lim, R. Yang, D. Li, and J. Song, "Signed power-of-two term allocation scheme for the design of digital filters," *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 577–584, May 1999.
[14] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Processing Mag.*, pp. 16–35, July 1992.
[15] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Computers*, vol. 8, pp. 330–334, Sept. 1959.
[16] J. S. Walther, "A unified algorithm for elementary functions," in *Spring Joint Computer Conf.*, 1971, pp. 379–385.
[17] Y. H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," *IEEE Trans. Computers*, vol. 42, pp. 99–102, Jan. 1993.
[18] C. S. Wu and A. Y. Wu, "Modified vector rotational CORDIC (MVR-CORDIC) algorithm and architecture," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 548–561, June 2001.
[19] ——, "A novel rotational VLSI architecture based on extended elementary-angle set CORDIC algorithm," in *Proc. IEEE 2nd IEEE Asia Pacific Conf. on ASICs*, Cheju, South Korea, 2000, pp. 111–114.

[20] ——, “A new trellis-based searching scheme for EEAS-based CORDIC algorithm,” in *Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing*, vol. 2, Salt Lake City, UT, 2001, pp. 1229–1232.

[21] G. D. Fornet, Jr., “The Viterbi algorithm,” *Proc. IEEE*, vol. 61, pp. 268–277, Mar. 1973.

An-Yeu (Andy) Wu received the B.S. degree from National Taiwan University in 1987, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, in 1992 and 1995, respectively, all in electrical engineering.

During 1987–1989, he served as a signal officer in the Army, Taipei, Taiwan, for his mandatory military service. During 1990–1995, he was a graduate teaching and research assistant with the Department of Electrical Engineering and Institute for Systems Research at the University of Maryland, College Park. From August 1995 to July 1996, he was a Member of Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ, working on high-speed transmission IC designs. From 1996 to July 2000, he was with the Electrical Engineering Department of National Central University, Taiwan. He is currently an Associate Professor with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taiwan. His research interests include low-power/high-performance VLSI architectures for DSP and communication applications, adaptive signal processing, and multirate signal processing.

Cheng-Shing (Benior) Wu was born in Taiwan, R.O.C., on November 25, 1973. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Central University, Taiwan, R.O.C., in 1996, 1997, and 2002, respectively.

Since 2001, he was a design engineer with Industrial Technology Research Institute (ITRI), Taiwan, R.O.C., engaged in the design of digital communication system. His research interests are in the areas of VLSI implementation of DSP algorithms, adaptive digital filter, and digital communication systems.