ON-LINE MSR-CORDIC VLSI ARCHITECTURE WITH APPLICATIONS TO COST-EFFICIENT ROTATION-BASED ADAPTIVE FILTERING SYSTEMS

Tzu-Hao Yu, Chi-Li Yu, Kai-Yuan Jheng, and An-Yeu (Andy) Wu

Graduate Institute of Electronics Engineering, and
Department of Electrical Engineering,
National Taiwan University, Taipei, 106, Taiwan, R.O.C.

Abstract- A novel on-line Mixed-Scaling-Rotation CORDIC (MSR-CORDIC) VLSI architecture is proposed. This architecture not only maintains the scaling-free property of the original MSR-CORDIC, but also achieves the target of on-line angle computation. Compared with other existing CORDIC solutions, the proposed architecture is faster and more cost-efficient, especially for QRD-RLS filtering systems. Moreover, this on-line MSR-CORDIC can also be adopted by other rotation-based DSP applications.

I. Introduction

The COordinate Rotational DiDigital Computer (CORDIC) algorithm [1] is a well-known hardware-efficient iterative algorithm for rotation-based arithmetic functions, such as Fast Fourier transformation (FFT), QRD-RLS filtering, Eigen Value Decomposition (EVD), and Singular Value Decomposition (SVD) [2]. Since CORDIC is computed by only a sequence of shift-and-add operations, it is much preferable to multiplier-based rotators. In the literature, many approaches were proposed to further improve the speed and reduce the hardware cost of the conventional CORDIC [3][4][5]. Among these techniques, the MSR-CORDIC [5] has been proved to be the most efficient vector rotational technique due to its low latency. It can successfully eliminate extra scaling iterations and merge them into the rotation iterations. It also has good numerical accuracy with the least iteration number for known rotational angles. Therefore, the MSR-CORDIC can be applied to the applications in which the angles to be rotated are known in advance, e.g. the twiddle factors of the FFT, and able to serve well in speed and accuracy with lower hardware complexity.

However, the MSR-CORDIC sacrifices a very important essence of the conventional CORDIC, angle computation (vectoring mode). The reason is that the originally proposed MSR-CORDIC in [5] searches for all possible elementary angles to either shorten the iteration sequences to pursue computational speed or to annihilate the scaling stages. As a result, the work of [5] cannot be adopted in angle-computation-required applications, such as QRD-RLS filtering [2].

In this paper, we present a novel on-line MSR-CORDIC, which is capable of conducting vectoring mode. Combined with the ordinary MSR-CORDIC, which can only execute rotation mode, the proposed on-line MSR-CORDIC can be applied to a QRD-RLS filter. Under the same convergence performance, the proposed architecture can save more than 30% of hardware cost compared to the conventional CORDIC. It also has the advantage of extremely few iterations, which leads to a very short latency. Compared with other existing solutions, the proposed on-line MSR-CORDIC is more cost-efficient and more suitable for high-speed applications.

II. Review of Existing CORDIC Algorithms

A. The Conventional CORDIC

Given a rotation angle $\Theta$, we can decompose it into several micro-rotations,

$$\Theta = \sum_{m=0}^{N-1} \mu \theta_m + \zeta$$

(1)
where $\theta$ are predefined elementary angles, $\tan^{-1}(2^{-n})$, for $n = 0, ..., N-1$; $N$ is the number of rotations; $\mu_n \in [-1,1]$ indicates the direction of rotation; $\zeta$ is the residual angle. Then, the iterative CORDIC algorithm can be written as Table I, where $z(n)$ is the summation of the micro-rotations; $P$ is the scaling factor.

Table I. Summary of the CORDIC algorithm.

**I. Rotational Phase**

For $n = 0, ..., N-1$

- **Perform** micro-rotation:

\[
\begin{bmatrix}
  x(n+1) \\
  y(n+1)
\end{bmatrix} =
\begin{bmatrix}
  1 & -\mu_n \times 2^{-n} \\
  \mu_n \times 2^{-n} & 1
\end{bmatrix}
\begin{bmatrix}
  x(n) \\
  y(n)
\end{bmatrix},
\]

(2)

- **Calculate** elementary angle

\[\theta_n = \mu_n \tan^{-1}(2^{-n}),\]

(3)

- **Update** accumulation angle

\[z(n+1) = z(n) + \mu_n \theta_n,\]

(4)

**End**

**II. Scaling Phase** (adjust norm)

\[
\begin{bmatrix}
  x_n \\
  y_n
\end{bmatrix} = P \begin{bmatrix}
  x(N) \\
  y(N)
\end{bmatrix},
\]

(5)

\[
P = \prod_{n=0}^{N-1} \left(1 + 2^{-n} \right).
\]

(6)

In the rotation mode of CORDIC, the expected rotation angle $\Theta$ is given. We usually set $z(0) = \Theta$ at the beginning. After $n$ iterations, we can obtain sequence $\mu_n$, which can be regarded as an alternative representation of $\Theta$, from the angle updating Eq. (4). On the contrary, $\Theta$ is not given in vectoring mode. In this case, we set $z(0)=0$ and $\mu_n = \text{sgn}(y(n))$ so that we can obtain sequence $\mu_n$ through running the CORDIC iterations. Then, we can learn the rotation angle $\Theta$ from sequence $\mu_n$.

In the conventional CORDIC, the iteration number is fixed. Hence, the scaling factor $P$ is a constant and can be known in advance. To compensate this, one can append few shift-and-add operations, instead of a generic multiplication, to the rotational phase.

In summary, the conventional CORDIC is very regular and suitable for not only rotation but also angle computation. However, it suffers from lengthy computation due to its large iteration number as well as compensation process for scaling factor. Hence, it is not favorable for high-speed applications.

**B. Mixed-Scaling-Rotation CORDIC (with only rotation mode) [5]**

Aiming at accelerating the conventional CORDIC for rotation mode, Lin and Wu [5] proposed the Mixed-Scaling-Rotation CORDIC. Firstly, the authors expand the signed power-of-two (SPT) terms in Eq. (2) to obtain an enlarged elementary angle set. Then, the elementary angle of Eq. (3), as well as the scaling factor of Eq. (6), is also modified. The MSR-CORDIC algorithm is summarized in Table II.

Table II. Summary of the MSR-CORDIC algorithm.

**Mixed Scaling Rotation**

For $n = 1, ..., N$

\[
\begin{bmatrix}
  x(n) \\
  y(n)
\end{bmatrix} =
\begin{bmatrix}
  \sum_{i=1}^{N} \eta_i(n) 2^{-i} \\
  \sum_{i=1}^{N} \mu_i(n) 2^{-i}
\end{bmatrix}
\begin{bmatrix}
  x(n-1) \\
  y(n-1)
\end{bmatrix},
\]

(7)

- **Calculate** elementary angle

\[\theta_n = \tan^{-1}\left(\frac{\sum_{i=1}^{N} \mu_i(n) 2^{-i}}{\sum_{i=1}^{N} \eta_i(n) 2^{-i}}\right),\]

(8)

- **Update** accumulation angle

\[z(n) = z(n-1) + \theta_n,\]

(9)

- **Amplifying factor in the n-th rotation**

\[p_n = \sqrt{1 + 2^{-2n}};\]

(10)

- **Product of the amplifying factors in the n-th rotation**

\[P_n = \prod_{i=1}^{n} p_i;\]

(11)

**End**

- **Scaling factor**

\[P = \prod_{n=1}^{N} p_n.\]

(12)

In Table II, $n$ denotes the n-th iteration and $N$ denotes the total number of iterations; $\mu_i(n), \eta_i(n) \in \{0,1,0,1\}$; $s_i(n), t_i(n) \in \{0,1,0,1\}$; and $S$ denotes the number of maximum shift. $I$ and $J$ denote the number of signed power-of-two (SPT) terms of $\sin \theta_n$ and $\cos \theta_n$, respectively; $\theta_n$ is the n-th elementary angle; $z(n)$ is the accumulative angle, and $z_0$ is $0$; $p_n$ denotes the product of the amplifying factors in the n-th iteration. The initial value of $P_0$ is 1, and $P$ denotes the Scaling Factor.

The principle of MSR-CORDIC algorithm is that it can perform the rotation and scaling operations at the same time. In the conventional and other existing CORDIC algorithms, the norm of a vector is always...
enlarged after a micro-rotation. On the contrary, Eq. (10) shows that the factor $p_k$ can be either greater or less than 1 in MSR-CORDIC algorithm. Therefore, the final scaling factor, $P$, can definitely be unity with a proper combination of micro-rotations. As a result, the additional scaling compensation can be eliminated. Moreover, Eq. (8) guarantees a much larger elementary angle set against other CORDIC algorithms. Therefore, the MSR-CORDIC can achieve any arbitrary angle in a very limited number of micro-rotations.

Combined with the features of the enlarged angle set and mixed-scaling mechanism, the MSR-CORDIC can dramatically reduce the iteration number. Finally, an extremely high speed can be achieved.

Nevertheless, the original MSR-CORDIC is dedicated to rotation mode. It is not suitable for the applications requiring on-line angle computation, i.e., vectoring mode. In this paper, we overcome these problems and propose a novel on-line MSR-CORDIC.

III. Proposed On-line MSR-CORDIC

A. The Algorithm of On-line MSR-CORDIC

The concept of the proposed on-line MSR-CORDIC can be illustrated in Fig. 1. Given that we need to rotate an initial vector by an arbitrary angle between 0 and 90 degrees, for iteration 1, we may have 8 elementary angles, $\frac{90}{8} \times 0, \frac{90}{8} \times 1, \ldots, \frac{90}{8} \times 7$. When we get a target angle, in iteration 1, we should rotate the initial vector by the elementary angle which is no larger than and closest to the target angle. In this case, as depicted in Fig. 1, our selection is $V_{13}$. After iteration 1, the difference between the target angle and the angle rotated is no larger than $\frac{90}{8}$. Hence for iteration 2, we further decompose the fundamental angle in iteration 1, $\frac{90}{8}$, into 8 elementary angles, $\frac{90}{8} \times 0, \frac{90}{8} \times 1, \ldots, \frac{90}{8} \times 7$. Again we rotate the vector by the elementary angle which is no larger than and closest to the target angle in iteration 2. In this case, our selection is $V_{26}$. After three iterations, the difference between the target angle and the angle rotated will be no larger than $\frac{90}{8 \times 8 \times 8} = 0.00039$. By this method, we can approach the target angle very fast, which means only a few iterations are required to attain the acceptable precision.

The aforementioned concept can be regarded as base-8 (octal) uniform angle quantization, just like the numerical octal quantization. Although this example works with base-8, in fact, any positive integer base is allowed, e.g., base-4, base-16, etc. Even non-uniform-base quantization, which means different bases are used in different iterations, can be adopted. In general, the elementary angle set of the on-line MSR-CORDIC algorithm can be defined as follows:

In iteration $k$, the elementary angles are:

$$0 \times \theta_f(k), 1 \times \theta_f(k), \ldots, [b(k) - 1] \times \theta_f(k)$$

(13)

where in iteration 1,

$$b(1) \times \theta_f(1) = 90^\circ$$

(14)

and for $k = 2, 3, \ldots, K$,

$$b(k) \times \theta_f(k) = \theta_f(k - 1)$$

(15)

where $K$ denotes the iteration number, $\theta_f(k)$ and $b(k)$ denote the fundamental angle and the base of the $k$-th iteration, respectively.

The angle estimation error of the proposed on-line MSR-CORDIC comes from two sources. The first is the difference between the target angle and the total angle rotated. This kind of error is, as mentioned above, limited by the fundamental angle of the last iteration, $\theta_f(K)$. The second source of error is that under the constraint of limited adder number, we cannot obtain all the desired elementary angles without error. These two kinds of error can both be reduced by increasing the hardware cost. Depending on the application under consideration, the design goal should be minimizing the hardware cost while meeting the system requirement.

Moreover, if we merely acquire the elementary angles without considering the scaling factors, the design will be unfeasible since we will not be able to control the overall scaling factor and need to compensate the final
norm dynamically. Therefore, two design constraints, as shown in the following equations, must be satisfied.

- **Constraint 1**
  
  For $k = 1, 2, \ldots, K$,
  
  $$ P_i(k) = P_i(k) = \ldots = P_{(i+1)}(k) = P(k), \tag{16} $$

- **Constraint 2**
  
  $$ \prod_{k=1}^{K} P(k) = 1, \tag{17} $$

where $P_i(k)$ denotes the scaling factor of the $i$-th elementary angle in the $k$-th iteration. Eq. (16) means that the elementary angles in a specific iteration must have the same scaling factor. Eqs. (16) and (17) together ensure that no matter which elementary angle is selected in each iteration, the overall scaling factor is unity. Hence the proposed on-line MSR-CORDIC is scaling-free and therefore needs no scaling factor compensation.

### B. Hardware Architectures

The iterative hardware architecture of the proposed on-line MSR-CORDIC in vectoring mode is shown in Fig. 2. Specifically, it serves as part of a boundary cell of a QRD-RLS adaptive filter, which will be mentioned in Section IV-A. Here we assume $I=2$, $J=2$, and base=16. Like conventional CORDIC, the objective is to rotate the vector $[x(n), y(n)]^T$ back to the $x$-axis such that the angle between them can be accrued, in which only the sign of $y(n+1)$ is concerned. However, since now the base is 16, we have 15 elementary angles to be back-rotated, in order to determine the division which the target vector belongs to. If our concern is to minimize the latency, we will need parallel architecture with $15 \times 3 = 45$ adders, where each 3-adder group is to obtain the sign of $y(n+1)$ when back-rotating one of the 15 elementary angles. On the other hand, if our concern is to minimize the hardware area, we will need serial architecture with $4 \times 3 = 12$ adders, where the four 3-adder groups are cascaded. Fig. 2 shows the architecture in-between, which is taken into comparison in Section IV-C.

The iterative hardware architecture of the proposed on-line MSR-CORDIC in rotation mode is shown Fig. 3. Specifically, this architecture serves as part of an internal cell of the QRD-RLS adaptive filter. Here we also assume that the design parameters is $I=2$, $J=2$, and base=16. The barrel shifter arrays and adders take the parameters from the boundary cell to rotate the vector $[x(n), y(n)]^T$ by one elementary angle out of sixteen candidates. Note that as the base increases, as long as the parameters $I$ and $J$ are fixed, the number of adders will not increase accordingly. This is advantageous since with larger base, we can approach the target angle faster. Although the cost of the boundary cell will increase, in the QRD-RLS filtering system, the number of boundary cells is only $O(M)$ while the number of internal cells is $O(M^2)$, where $M$ is the tap number of the filter. That is to say, internal cells dominate the hardware cost of a QRD-RLS filter.

### IV. Simulation Results

#### A. Simulation Environment

Fig. 4 shows the block diagram of the simulation environment [7]. The random signal applied to the channel input consists of a Bernoulli sequence. The impulse response of the channel is described by the raised cosine,
where the parameter $W$ controls the eigenvalue spread of the correlation matrix of the tap inputs in the equalizer. The AWGN block produces white Gaussian noise with zero mean and variance 0.001. The equalizer is a QRD-RLS adaptive filter containing a systolic array, as shown in Fig. 5, for coefficient-calculation. Usually, an on-line vectoring-mode CORDIC is adopted as a boundary cell, and a rotation-mode CORDIC is adopted as an internal cell in the systolic array. In the following simulations, the tap-number is set to 11.

**B. Results of the Learning Curves**

**Simulation I: QRD-RLS with Double-Precision Multipliers/Dividers**

Fig. 6 shows the learning curves with the internal and boundary cells of the equalizer constructed of double-precision multipliers/dividers. For each eigenvalue spread, an approximation to the ensemble-average learning curve of the adaptive equalizer is obtained by averaging the instantaneous-squared-error curve over 200 independent trials of the computer experiment. The steady-state value of the average squared error with $W=3.5$ is 0.0044, which can be viewed as a reference value. When we replace the internal and boundary cells with other CORDIC processors, the minimum hardware costs of them which can approximate this reference value will be compared.

**Simulation II: QRD-RLS with the conventional CORDIC**

Fig. 7 shows the learning curves when we replace the internal and boundary cells with conventional CORDIC processing elements. The iteration number, $N$, of the CORDIC operation ranges from 5 to 8. As the figure shows, when the iteration number equals 5 or 6, due to the large error induced by the processing elements, the filter does not converge to a steady state. On the other hand, for the cases when $N=7$ and $N=8$, the steady-state values of the average squared error are 0.0064 and 0.0046, respectively. Also, the convergence rates of both cases are similar to the case in simulation I. That is to say, the conventional CORDIC processors with 8 iterations are sufficient under this simulation environment. The hardware cost of the conventional CORDIC with $N=8$ will be taken into comparison in the next section.

**Simulation III: QRD-RLS with the proposed on-line MSR-CORDIC**

Fig. 8 shows the learning curves when we replace the internal and boundary cells with the proposed on-line MSR-CORDIC.
MSR-CORDIC processing elements. Here the iteration number $K$ equals 2 in the processing element. The base in each of the two iterations is 16. The parameters $I$ and $J$ are both 2. As can be seen, the convergence rates of the learning curves are similar to the case in simulation I. The steady-state value of the average squared error with $W=3.5$ is 0.0045, which is also similar to the case in simulation I. The hardware cost of the on-line MSR-CORDIC with the specified design parameters will be compared with other works in the next section.

![Fig. 8. Learning curves of the QRD-RLS algorithm using the proposed on-line MSR-CORDIC processing elements.](image)

**C. Comparison results**

Based on the same performance of convergence in 11-tap QRD-RLS, the proposed on-line MSR-CORDIC is compared with the conventional CORDIC and another scaling-free solution [6]. To achieve the high-speed requirement, expanded architectures are selected instead of iterative ones in the comparisons. Table III shows the comparison of hardware complexity. In each iteration, the conventional CORDIC and [6] require only 2 adders, and MSR-CORDIC requires 18 and 6 adders in vectoring and rotation mode respectively. However, MSR-CORDIC has the smallest number of adders due to its extremely few iterations. Moreover, when the tap-number increases, e.g. 30, MSR-CORDIC can save more than 30% number of adders.

Table IV shows the comparison of latency. Due to its extremely few iterations, the total latency of MSR-CORDIC is the shortest among the three schemes.

**V. Conclusions**

In this paper, a novel on-line MSR-CORDIC has been presented. This architecture not only maintains the scaling-free property of the original MSR-CORDIC, but also achieves on-line angle computation. Compared with the other solutions, this architecture is faster and more cost-efficient. In addition to QRD-RLS filter, it can be applied to any other rotation-based DSP applications.

**References**


