

Merged Scaling Multiplication CORDIC Algorithm

Shaoyun Wang
Crystal Semiconductor Corp.
4210 S. Industrial Dr.
Austin, TX 78744, USA
shaoyun@pine.ece.utexas.edu

Vincenzo Piuri
Dept. of Elec. and Info.
Politecnico di Milano
20133 Milano, Italy
piuri@elet.polimi.it

Earl E. Swartzlander, Jr.
Dept. of Elec. and Com. Eng.
The University of Texas
Austin, TX 78712, USA
e.swartzlander@compmail.com

Abstract— In an N -bit fixed-point processor, the COordinate Rotation DIGital Computer (CORDIC) algorithm usually consists of N elementary iterations and one scaling multiplication. For the Constant Scale Factor Algorithm, we propose a new approach that merges the iterations with the scaling to achieve a more regular and pipelinable structure of the processor, which is well suited for VLSI implementation.

I. INTRODUCTION

The CORDIC algorithm is a well-known and widely-studied iterative technique ([1] and [2]) for evaluating many basic arithmetic operations and mathematical functions. Examples are multiplication, division, square root, sine, cosine, inverse tangent, hyperbolic sine, hyperbolic cosine, and inverse hyperbolic tangent. The results of these functions can be exploited to generate other transcendental functions such as tangent, hyperbolic tangent, logarithms, and exponentials.

These functions are widely used in several massive-computing applications (e.g., dynamic system modeling, control, robotics, computer graphics, digital signal processing, image processing, document processing, imaging, simulation, and virtual reality). Firstly, CORDIC can be exploited to realize mathematical co-processors having higher precision than other approaches. In addition, several algorithms in the above applications may be expressed in terms of basic plane rotations.

Many papers have been published on the design of advanced architectural structures for CORDIC processors to enhance the performance or to reduce the circuit complexity (e.g., see [3]- [5]). Other approaches reduce the system latency or increase the throughput by adopting general digital-design techniques, e.g., by cascading the CORDIC stages to achieve a combinatoric structure [6] or by pipelining this cascade [7]. In [8] two families of processors, the Combined and the Pipelined CORDIC pro-

cessors, are presented by performing more than one CORDIC iteration in a single clock cycle or in a single pipeline clock cycle, respectively. This provides a variety of options to the designer who can choose the best trade-off among circuit complexity, clock cycle time, latency and throughput according to the requirements of the specific application.

Almost all the papers in the literature consider the scaling multiplication of the iterations' result by the scale factor as an overhead of the algorithm and separate it from the iterations. This leads to an irregular structure of the processor and, as a consequence, it may introduce some architectural problems. For example, the separate multiplier is often under-used with respect to the other components. Additional problems occur when a very regular design is required to deal with fault tolerance issues; in these cases, in fact, an array of identical modules is highly desirable in order to allow for an easy error detection, fault localization, and reconfiguration without requiring a separate modular redundancy approach for the multiplier.

When only a function generator is envisioned, it is trivial to deal with the scale factor in the case of the Constant Scale Factor CORDIC Algorithm. The desired function of an unknown angle θ is in fact generated by rotating a constant predefined vector by the angle θ and, then, by reading the coordinates of the rotated vector. Instead of using the standard initial vector (1,0), we can replace it with the constant vector whose coordinates have been pre-scaled by the given Constant Scale Factor.

In this paper, for the Constant Scale Factor Algorithm, we propose a new approach that solves the scaling problem in the general case by directly merging the CORDIC iterations with the scaling multiplication. The scale factors is decomposed into multiple factors and each of these factor is merged with one iteration. The result generated by the application of the modified iterations is the correctly scaled result expected at the end of the whole traditional approach (including the scaling multiplication). No additional

manipulation is required. The structures that can be derived to implement this approach have a high regularity with respect to the traditional ones and, as a consequence, they may be more attractive when regularity and modularity are design issues. For example, they may be better suited for pipelining and fault tolerance.

Section 2 summarizes the CORDIC operation. Section 3 analyzes the numeric structure of the Constant Scale Factor for the scaling multiplication and shows the factorization technique. Section 4 exploits this result by proposing a method to merge the scaling multiplication with the CORDIC iterations: the new algorithm is called Merged Scaling Multiplication CORDIC algorithm.

II. THE TRADITIONAL CORDIC ALGORITHM

The CORDIC algorithm is a bit-recursive implementation of the forward and backward Givens rotations [2], also called *rotation* and *vectoring*, respectively.

Consider the vector $\mathbf{v}$ in the plane xy , having a vertex at the real coordinates x^* and y^* . Let z^* be the rotation angle of such a vector with respect to the x axis. The CORDIC equations at the i -th step are:

$$\begin{cases} x_{i+1} = k_{m,i}(x_i - m\sigma_i 2^{-S(m,i)} y_i) \\ y_{i+1} = k_{m,i}(y_i + \sigma_i 2^{-S(m,i)} x_i) \\ z_{i+1} = z_i - \sigma_i \alpha_{m,i} \end{cases} \quad (1)$$

where the iterations are repeated $\forall i, i = 0, 1, 2, \dots, N-1$. The *coordinate parameter* m identifies the kind of coordinate system, namely, circular, linear, and hyperbolic coordinates for m equal to 1, 0, and -1, respectively. The *rotation direction* σ_i for rotation is $\sigma_i = \text{sign}(z_i)$, while it is $\sigma_i = \text{sign}(x_i \cdot y_i)$ for vectoring. The *shift sequence* $S(m,i)$ (i.e., the exponent of the rotation coefficient $2^{S(m,i)}$) is defined by $S(m,i) = 0, 1, 2, 3, 4, 5, \dots$ for $m = 1$, $S(m,i) = 1, 2, 3, 4, 5, 6, \dots$ for $m = 0$, and $S(m,i) = 1, 2, 3, 4, 5, \dots$ for $m = -1$ (where items equal to $\frac{3^{i+2}-1}{2}$ are repeated twice in the sequence). The *rotation angle* $\alpha_{m,i}$ is given by $\alpha_{m,i} = \frac{1}{\sqrt{m}} \arctan(\sqrt{m} 2^{-S(m,i)})$. The *scale factor* $k_{m,i}$ is required to correct the distortion introduced by the linearized rotation in the x and y coordinates; with the above assumptions, it is $k_{m,i} = \cos \alpha_{m,i} = \sqrt{1 + m\sigma_i^2 2^{-2S(m,i)}}$. After all the iterations, the total scale factor is $K_m = \prod_{i=0}^{N-1} k_{m,i} = \prod_{i=0}^{N-1} \cos \alpha_{m,i} = \prod_{i=0}^{N-1} \sqrt{1 + m\sigma_i^2 2^{-2S(m,i)}}$ (the product starts from $i = 1$ for $m = -1$).

When rotation of the vector $\mathbf{v}$ by the angle θ must be performed, we set $x_0 = x^*$, $y_0 = y^*$, and $z_0 = \theta$;

at the end of all iterations, the vertex of the rotated vector $\mathbf{v}'$ has coordinates (x_N, y_N) . If vectoring is required, we set $x_0 = x^*$, $y_0 = y^*$, and $z_0 = 0$; at the end of the iterations, the required angle z^* is equal to z_N , while the modulus $|\mathbf{v}|$ is given by x_N . By setting the suited initial values of x^* and y^* , we can compute the value of the trigonometric and transcendent elementary functions.

In traditional Constant Scale Factor CORDIC, the rotation directions are restricted to $\sigma_i = \{-1, 1\}$. In this case, the scale factor becomes constant with respect to the rotation coefficients, and depends only on the word length N and the coordinate parameter m . A constant scale factor simplifies the correction since it needs to be applied only once: either to the final result or to the initial operands (as in the traditional approaches), or spread along the CORDIC iterations (as we are proposing in this paper).

To enhance the convergence speed of the algorithm, some authors [9] [10] prefer to use also void rotations, i.e., they consider 0 as an acceptable value for σ_i . However, this implies that the final scale factor is not constant but depends on the specific sequence of rotations actually applied. If void rotation steps are obtained by using micro-rotations [5], the above total scaling factor is preserved so as correction still need to be applied only once, but the number of basic micro-iterations is twice the number of traditional iterations (i.e., additions are doubled).

III. THE DECOMPOSITION OF THE CONSTANT SCALE FACTOR

In a CORDIC processor having finite precision equal to N bits, consider the product $H_m(N-1)$ given by:

$$H_m(N-1) = \prod_{j=0}^{N-1} (1 + h_{m,j} 2^{-j}) \quad (2)$$

where the coefficients $h_{m,j}$ are in $\{-1, 1\}$.

We define the *H-decomposition* $H_m(N-1)$ of the total scale factor $K_m(N-1)$ as the above product when there is a set of coefficients $\{h_{m,j} \mid h_{m,j} \in \{-1, 1\}\}$ so that:

$$\frac{1}{K_m(N-1)} = H_m(N-1) \quad (3)$$

in the given N -bit precision. The coefficients $h_{m,j}$ of the H-decomposition are called *H-factors*. The ordered sequence $\{h_{m,0}, h_{m,1}, h_{m,2}, \dots, h_{m,N-1}\}$ of H-factors is called the *H-sequence*.

Eq.3 confirms that the H-decomposition can be used instead of the inverse of the total scale factor to correctly scale the results generated by the CORDIC

algorithm. Stating that Eq. 3 holds for N -bit precision requires that:

$$\left| H(N-1) - \frac{1}{K(N-1)} \right| < 2^{-(N-1)} \quad (4)$$

In other words, the N most significant bits of the representation of $H_m(N-1)$ coincides with the N most significant bits of the representation of scale correction $\frac{1}{K_m(N-1)}$.

The decomposition of Eq. 2 is not feasible for any real number. However, we have found that this decomposition exists for the total scale factors both in the circular and in the hyperbolic coordinates, as stated in the following theorem (the complete proof is given in [11]):

Theorem 1 *In the CORDIC algorithm having N -bit precision, the H -decomposition $H(N-1)$ of the total scale factor $K(N-1)$ for the circular and the hyperbolic coordinates always exists in that precision.*

A consequence of all the possible cases considered in the proof of Theorem 1 is the following rule to iteratively build the H -sequence for any i -bit precision in the circular coordinates. If the precision is 1 bit then the H -sequence is $\{-1\}$. Otherwise, given the H -sequence for the precision $i-1$, the H -sequence for $H(i)$ in the precision i is generated by the first of the following cases that satisfies the Eq. 4:

- $\{h_j \mid h_j \in H\text{-sequence of } H(i-1) \text{ for } j = 0, \dots, i-1; h_i = -1\}$,
- $\{h_j \mid h_j \in H\text{-sequence of } H(i-1) \text{ for } j = 0, \dots, i-2; h_{i-1} = 1; h_i = -1\}$,
- $\{h_j \mid h_j \in H\text{-sequence of } H(i-1) \text{ for } j = 0, \dots, i-1; h_i = 1\}$,
- $\{h_j \mid h_j \in H\text{-sequence of } H(i-1) \text{ for } j = 0, \dots, i-2; h_{i-1} = -1; h_i = 1\}$.

Similar results can be achieved also for the hyperbolic coordinates.

IV. THE MERGED SCALING MULTIPLICATION CORDIC ALGORITHM

The CORDIC algorithm can be written in the form of matrix multiplications representing the partial rotations associated with the ATR coefficients:

$$\begin{pmatrix} x_N \\ y_N \end{pmatrix} = \begin{pmatrix} \cos \alpha_{m,0} & -\sigma_0 \sin \alpha_{m,0} \\ \sigma_0 \sin \alpha_{m,0} & \cos \alpha_{m,0} \end{pmatrix} \times \begin{pmatrix} \cos \alpha_{m,1} & -\sigma_1 \sin \alpha_{m,1} \\ \sigma_1 \sin \alpha_{m,1} & \cos \alpha_{m,1} \end{pmatrix} \times \dots \times \begin{pmatrix} \cos \alpha_{m,N-1} & -\sigma_{N-1} \sin \alpha_{m,N-1} \\ \sigma_{N-1} \sin \alpha_{m,N-1} & \cos \alpha_{m,N-1} \end{pmatrix} \times \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

By extracting the Constant Scale Factor in order to obtain a simpler description in the tangent functions only, we derive:

$$\begin{pmatrix} x_N \\ y_N \end{pmatrix} = \frac{1}{K_m} \times \begin{pmatrix} 1 & -\sigma_0 \tan \alpha_{m,0} \\ \sigma_0 \tan \alpha_{m,0} & 1 \end{pmatrix} \times \begin{pmatrix} 1 & -\sigma_1 \tan \alpha_{m,1} \\ \sigma_1 \tan \alpha_{m,1} & 1 \end{pmatrix} \times \dots \times \begin{pmatrix} 1 & -\sigma_{N-1} \tan \alpha_{m,N-1} \\ \sigma_{N-1} \tan \alpha_{m,N-1} & 1 \end{pmatrix} \times \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

Consider first the case of the circular coordinate system. From the definitions given in Section 2, it is $\tan \alpha_{1,i} = 2^{-i}$. As a consequence, we derive:

$$\begin{pmatrix} x_N \\ y_N \end{pmatrix} = K_m \times \begin{pmatrix} 1 & -\sigma_0 2^{-0} \\ \sigma_0 2^{-0} & 1 \end{pmatrix} \times \begin{pmatrix} 1 & -\sigma_1 2^{-1} \\ \sigma_1 2^{-1} & 1 \end{pmatrix} \times \dots \times \begin{pmatrix} 1 & -\sigma_{N-1} 2^{-(N-1)} \\ \sigma_{N-1} 2^{-(N-1)} & 1 \end{pmatrix} \times \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

By replacing the Constant Scale Factor K_m with its factorized representation as discussed in Section 3, we obtain:

$$\begin{pmatrix} x_N \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & -\sigma_0 2^{-0} \\ \sigma_0 2^{-0} & 1 \end{pmatrix} \times (1 + h_{N-1} 2^{-(N-1)}) \begin{pmatrix} 1 & -\sigma_1 2^{-1} \\ \sigma_1 2^{-1} & 1 \end{pmatrix} \times \dots \times (1 + h_1 2^{-1}) \begin{pmatrix} 1 & -\sigma_{N-1} 2^{-(N-1)} \\ \sigma_{N-1} 2^{-(N-1)} & 1 \end{pmatrix} \times (1 + h_0 2^{-0}) \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 1 & -\sigma_0 2^{-0} \\ \sigma_0 2^{-0} & 1 \end{pmatrix} \times \begin{pmatrix} 1 + h_{N-1} 2^{-(N-1)} & -\sigma_1 2^{-1} - \sigma_1 h_{N-1} 2^{-N} \\ \sigma_1 2^{-1} + \sigma_1 h_{N-1} 2^{-N} & 1 + h_{N-1} 2^{-(N-1)} \end{pmatrix} \times \dots \times (1 + h_0 2^{-0}) \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

It is worth noting that the terms $\sigma_i h_{N-i} 2^{-N}$ contribute only to accumulating error since the processor's word length is N . Therefore, these terms can

be removed without introducing any additional approximation error in the evaluation of the CORDIC result. We obtain:

$$\begin{pmatrix} x_N \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & -\sigma_0 2^{-0} \\ \sigma_0 2^{-0} & 1 \end{pmatrix} \times \begin{pmatrix} 1 + h_{N-1} 2^{-(N-1)} & -\sigma_1 2^{-1} \\ \sigma_1 2^{-1} & 1 + h_{N-1} 2^{-(N-1)} \end{pmatrix} \times \cdots \times \begin{pmatrix} 1 + h_1 2^{-1} & -\sigma_{N-1} 2^{-(N-1)} \\ \sigma_{N-1} 2^{-(N-1)} & 1 + h_1 2^{-1} \end{pmatrix} \times (1 + h_0 2^{-0}) \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

The iteration equations for the Merged Scaling Multiplication CORDIC Algorithm are finally:

$$\begin{cases} x_{i+1} = x_i + h_{N-i} 2^{-N+i} x_i - m \sigma_i 2^{-S(m,i)} y_i \\ y_{i+1} = y_i + h_{N-i} 2^{-N+i} y_i + \sigma_i 2^{-S(m,i)} x_i \\ z_{i+1} = z_i - \sigma_i \alpha_{m,i} \end{cases}$$

V. CONCLUSION

The merged scaling multiplication CORDIC algorithm decomposes the final scale factor into each iteration stage. The multiplication becomes a shift (by a predefined number of bits) and add. It is possible to verify that the circuit complexity and the latency of the architectures to implement this approach are practically identical to those of the traditional approach.

The main advantage provided by this approach with respect to the traditional architectures is the higher regularity of the structures that can be designed to implement it. In fact, now it is not necessary to cascade a constant multiplier to the CORDIC core to achieve the final adjusted result. This avoids the irregularities in the data flow due to the use of two different structures (namely, the CORDIC core and the scaling multiplier). This characteristic is attractive when high regularity is a mandatory requirement for the implementation. For example, it is very attractive for a highly regular pipelined architecture. When mission-critical applications are concerned, it is useful to support efficiently the use of testing techniques requiring built-in self-test components as well as the introduction of fault tolerant techniques for concurrent error detection and correction (e.g., data coding, time redundancy) or for reconfiguration of the interconnections between components.

REFERENCES

- [1] Jack E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers*, Vol. EC-8, 1959, pp. 330-334.
- [2] John S. Walther, "A Unified Algorithm For Elementary Functions," *Spring Joint Computer Conference Proceedings*, Vol. 38, 1971, pp. 379-385.
- [3] Jean-Marc Delosme, "VLSI Implementation of Rotations in Pseudo Euclidean Spaces," *Proceedings of the 1983 IEEE International Conference on Acoustic, Speech, and Signal Processing*, Vol. 2, 1983, pp. 927-930.
- [4] R. Kundmund, et al., "CORDIC Processor with Carry Save Architecture," *Proceedings of the 1990 European Solid State Circuits Conference*, Grenoble, France, September 1990, pp. 193-196.
- [5] Naofumi Takagi, Tohru Asada, and Shuzo Yajima, "Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation," *IEEE Transaction on Computers*, Vol. 40, 1991, pp. 989-995.
- [6] Dirk Timmermann, Helmut Hahn, and Bedrich J. Hosticka, "Low Latency Time CORDIC Algorithms," *IEEE Transactions on Computers*, Vol. 41, 1992, pp. 1010-1015.
- [7] E. F. Deprettere, P. Dewilde, and R. Udo, "Pipelined CORDIC Architectures for Fast VLSI Filtering and Array Processing," *IEEE Transaction on Signal Processing*, Vol. 40, 1992, pp. 834-844.
- [8] Shaoyun Wang, Vincenzo Piuri and Earl E. Swartzlander, Jr., "A Unified View of CORDIC Processor Design," *Proceedings of the 39th Midwest Symposium on Circuits and Systems*, Ames, IA, August 1996.
- [9] Miloš D. Ercegovic and Tomás Lang, "Redundant and On-Line CORDIC: Application to Matrix Triangularization and SVD," *IEEE Transaction on Computers*, Vol. 39, 1990, pp. 725-740.
- [10] Shaoyun Wang and Earl E. Swartzlander, Jr., "Critically Damped CORDIC Algorithm," *Proceedings of the 37th Midwest Symposium on Circuits and Systems*, Lafayette, LA, pp. 253-256, August 1994.
- [11] Shaoyun Wang, Vincenzo Piuri and Earl E. Swartzlander, Jr., "Merged Scaling Multiplication CORDIC Algorithm," Internal Report #96-140, Department of Electronics and Information, Politecnico di Milano, 20133 Milano, Italy, August 1996.