Review Paper on Cordic Algorithms

Minakshi Dahake[1], Prof. Amol Boke[2]
Department of Electronics & Communication Engineering
[1]MTech,GHRAET, Maharashtra,India
[2]Prof GHRAET, Maharashtra,India

Abstract-*Year 2009 marks the completion of 50 years of the invention of CORDIC (Coordinate Rotation Digital Computer) by Jack E. Volder. The beauty of CORDIC lies in the fact that by simple shift-add operations, it can perform several computing tasks such as the calculation of trigonometric, hyperbolic and logarithmic functions, real and complex multiplications, division, square-root, solution of linear systems, eigenvalue estimation, singular value decomposition, QR factorization and many others. As a consequence, CORDIC has been utilized for applications in diverse areas such as signal and image processing, communication systems, robotics and 3-D graphics apart from general scientific and technical computation. In this article, we present a brief overview of the key developments in the CORDIC algorithms and architectures along with their potential and upcoming applications..*

Keywords-Arithmetic circuits, CORDIC, CORDIC algorithms, digital signal processing chip, VLSI..

I. INTRODUCTION

COORDINATE Rotation Digital Computer is abbreviated as CORDIC. The key concept of CORDIC arithmetic is based on the simple and ancient principles of two-dimensional geometry. But the iterative formulation of a computational algorithm for its implementation was first described in 1959 by Jack E. Volder [1], [2] for the computation of trigonometric functions, multiplication and division. This year therefore marks the completion of 50 years of the CORDIC algorithm. Not only a wide variety of applications of CORDIC have emerged in the last 50 years, but also a lot of progress has been made in the area of algorithm design and development of architectures for high-performance and low-cost hardware solutions of those applications. CORDIC-based computing received increased attention in 1971, when John Walther [3], [4] showed that, by varying a few simple parameters, it could be used as a single algorithm for unified implementation of a wide range of elementary transcendental functions involving logarithms, exponentials, and square roots along with those suggested by Volder [1]. During the same time, Cochran [5] benchmarked various algorithms, and showed that CORDIC technique is a better choice for scientific calculator applications. The popularity of CORDIC was very much enhanced thereafter primarily due to its potential for efficient and low-cost implementation of a large class of applications which include: the generation of trigonometric, logarithmic and transcendental elementary functions; complex number multiplication, eigenvalue computation, matrix inversion, solution of linear systems and singular value decomposition (SVD) for signal processing, image processing, and general scientific computation. Some other popular and upcoming applications are:

1) direct frequency synthesis, digital modulation and coding for speech/music synthesis and communication;
2) direct and inverse kinematics computation for robot manipulation;
3) planar and three-dimensional vector rotation for graphics and animation.

Although CORDIC may not be the fastest technique to perform these operations, it is attractive due to the simplicity of its hardware implementation, since the same iterative algorithm could be used for all these applications using the basic shift-add operations of the form a±b.2-i.

Keeping the requirements and constraints of different application environments in view, the development of CORDIC algorithm and architecture has taken place for achieving high throughput rate and reduction of hardware-complexity as well as the latency of implementation. Some of the typical approaches for reduced-complexity implementation are focused on minimization of the complexity of scaling operation and the complexity of barrel-shifter in the CORDIC engine. Latency of implementation is an inherent drawback of the conventional CORDIC algorithm. Angle recoding schemes, mixed-grain rotation and higher radix CORDIC have been developed for reduced latency realization. Parallel and pipelined CORDIC have been suggested for high-throughput computation. The objective of this article is not to present a detailed survey of the developments of algorithms, architectures and applications of CORDIC, which would require a few doctoral and masters level dissertations. Rather we aim at providing the key developments in algorithms and architectures alongwith an overview of the major application areas and upcoming applications. We shall however discuss here the basic principles of CORDIC operations for the benefit of general readers.

II. BASIC CORDIC TECHNIQUES

In this Section, we discuss the basic principle underlying the CORDIC-based computation, and present its iterative algorithm for different operating modes and planar coordinate systems. At the end of this section, we discuss the extension of two-dimensional rotation to multidimensional formulation.

The CORDIC Algorithm

As shown in Fig. 1, the rotation of a two-dimensional vector $P_0=[x_0\ y_0]$an angle Θ, to obtain a rotated vector $P_n = [\ x_n y_n]$ could be performed by the matrix product $Pn = R_{P0}$, where R is the rotation matrix:

$$R = \begin{bmatrix} Cos\theta & -Sin\theta \\ Sin\theta & Cos\theta \end{bmatrix} \quad (1)$$

By factoring out the cosine term in (1), the rotation matrix R can be rewritten as,

$$R=\left[(1 + tan^2\theta)^{-1/2}\right]\begin{bmatrix} 1 & -tan\theta \\ tan\theta & 1 \end{bmatrix} \quad (2)$$

and can be interpreted as a product of a scale-factor $K = \left[(1 + tan^2\theta)^{-1/2}\right]$with a pseudorotation matrix R_c, given by

$$Rc=\begin{bmatrix} 1 & -tan\theta \\ tan\theta & 1 \end{bmatrix} \quad (3)$$

The pseudo rotation operation rotates the vector P0 by an angle Θ and changes its magnitude by a factor K= cosΘ, to produce a pseudo-rotated vector $P_n' = R_c P_0$.

To achieve simplicity of hardware realization of the rotation, the key ideas used in CORDIC arithmetic are to (i) decompose the rotations into a sequence of elementary rotations through predefined angles that could be implemented with minimum hardware cost; and (ii) to avoid scaling, that might involve arithmetic operation, such as square-root and division. The second idea is based on the fact the scale-factor contains only the magnitude information but no information about the angle of rotation.

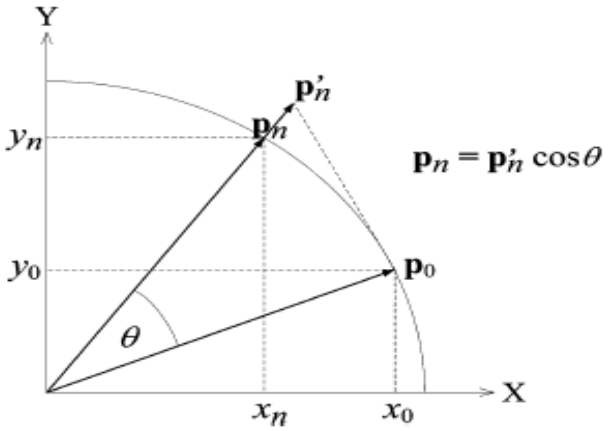

Figure 1: Rotation of vector on a two-dimensional plane.

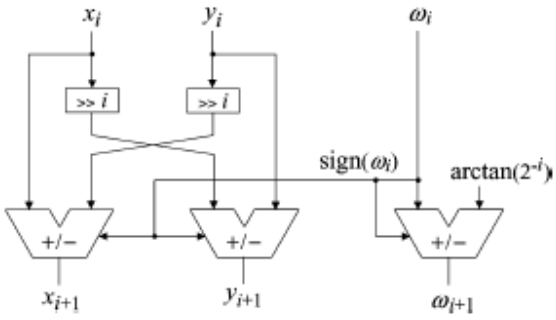

Figure 2: Hardware implementation of a CORDIC iteration.

TABLE I GENERALIZED CORDIC ALGORITHM

m	Rotation Mode	Vectoring Mode
0	$x_n = K\ (x_0 Cos\omega_0 - y_0 sin\omega_0)$	$x_n = K\sqrt{x_0^2 + y_0^2}$
	$y_n = K\ (y_0 Cos\omega_0 + y_0 sin\omega_0)$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + \tan^{-1}\left(\frac{y_0}{x_0}\right)$
1	$x_n = x_0$	$x_n = x_0$
	$y_n = y_0 + x_0\omega_0$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + \left(\frac{y_0}{x_0}\right)$
-1	$x_n = K_h\ (x_0 Cosh\omega_0 - y_0 sinh\omega_0)$	$x_n = K_h\sqrt{x_0^2 - y_0^2}$
	$y_n = K_h\ (y_0 Cosh\omega_0 + y_0 sinh\omega_0)$	$y_n = 0$
	$\omega_n = 0$	$\omega_n = \omega_0 + \tanh^{-1}\left(\frac{y_0}{x_0}\right)$

Generalization of the CORDIC Algorithm

In 1971, Walther found how CORDIC iterations could be modified to compute hyperbolic functions [3] and reformulated the CORDIC algorithm in to a generalized and unified form which is suitable to perform rotations in circular, hyperbolic and linear coordinate systems. The unified formulation includes a new variable m, which is assigned different values for different coordinate systems. The generalized CORDIC is formulated as follows:

$$x_{i+1} = x_i - m\sigma_i . 2^{-i} . y_i$$
$$y_{i+1} = y_i - \sigma_i . 2^{-i} . x_i$$
$$\omega_{i+1} = \omega_i - \sigma_i . \alpha_i \quad (4)$$

Multidimensional CORDIC

The CORDIC algorithm was extended to higher dimensions using simple Householder reflection [7]. The Householder reflection matrix is defined as

$$H_m = I_m^{-2} \cdot \frac{uu^T}{uu} \quad (5)$$

III. ADVANCED CORDIC ALGORITHMS AND ARCHITECTURES

CORDIC computation is inherently sequential due to two main bottlenecks: 1) the micro-rotation for any iteration is performed on the intermediate vector computed by the previous iteration and 2) the (i+1)th iteration could be started only after the completion of the ith iteration, since the value of σ_{i+1}which is required to start the th iteration could be known only after the completion of the th iteration. To alleviate the second bottleneck some attempts have been made for evaluation of values corresponding to small micro-rotation angles [9], [10]. However, the CORDIC iterations could not still be performed in parallel due to the first bottleneck. A partial parallelization has been realized in [11] by combining a pair of conventional CORDIC iterations into a single merged iteration which provides better area-delay efficiency. But the accuracy is slightly affected by such merging and cannot be extended to a higher number of conventional CORDIC iterations since the induced error becomes unacceptable [11]. Parallel realization of CORDIC iterations to handle the first bottleneck by direct unfolding of micro-rotation is possible, but that would result in increase in computational complexity and the advantage of simplicity of CORDIC algorithm gets degraded [12], [13]. Although no popular architectures are known to us for fully parallel implementation of CORDIC, different forms of pipelined implementation of CORDIC have however been proposed for improving the computational throughput [14].

To handle latency bottlenecks, various techniques have been developed and reported in the literature. Most of the well known algorithms could be grouped under, high-radix CORDIC, the angle-recoding method, hybrid micro-rotation scheme, redundant CORDIC and differential CORDIC which we discuss briefly in the following subsections

Parallel Angle Recoding

The AR methods [19], [21] could be used to reduce the number of iterations by more than 50%, when the angle of rotation is known in advance. However, for unknown rotation angles, their hardware implementation involves more cycle time than the conventional implementation, which results in a reduction in overall efficacy of the algorithm. To reduce the cycle time of CORDIC iterations in such cases, a parallel angle selection scheme is suggested in [22], which can be used in conjunction with the AR method, to gain the advantages of the reduction in iteration count, without further increase in the cycle time. The parallel AR scheme in [22] is based on dynamic angle selection, where the elementary angles can be tested in parallel and the direction for the micro-rotations can be determined quickly to minimize the iteration period.

Implementation of Hybrid CORDIC

To derive the efficiency of hybrid CORDIC, the coarse and fine rotations are performed by separate circuits as shown in Fig. 5. The coarse rotation phase is performed by the CORDIC processor-I and the fine rotation phase is performed by CORDIC processor-II. To have fast implementation, processor-I performs a pair of ROM look-up operations followed by addition to realize the rotation through angle . Since could be expressed as a linear combination of angels of small enough magnitude , where , the computation of fine rotation phase can be realized by a sequence of shift-and-add operations. For implementation of the fine rotation phase, no computations are involved to decide the direction of micro-rotation, since the need of a micro-rotation is explicit in the radix-2 representation of . The radix-2 representation could also be recoded to express where as shown in [9]. Since the direction of micro-rotations are explicit in such a representation of , it would be possible to implement the fine rotation phase in parallel for low-latency realization. The hybrid decomposition could be used for reducing the latency by ROM-based realization of coarse operation. This can also be used for reducing the hardware complexity of fine rotation phase since there is no need to find the direction of microrotation. Several options are however possible for the implementation of these two stages. A form of hybrid CORDIC is suggested in [23] for very-high precision CORDIC rotation where the ROM size is reduced to nearly bits. The coarse rotations could be implemented as conventional CORDIC through shift-add operations of micro-rotations if the latency is tolerable.

Parallel CORDIC Based on Coarse-Fine Decomposition

In [31], the authors have proposed two angle recoding techniques for parallel detection of direction of micro-rotations, namely the binary to bipolar recoding (BBR) and micro-rotation angle recoding (MAR) to be used for the coarse part of the input angle . BBR is used to obtain the polarity of each bit in the radix-2 representation of to determine the rotation direction. MAR is used to decompose each positional binary weight into a linear combination of arctangent terms. It is further shown in [32] that, the rotation direction can be decided once the input angle is known to

enable parallel computation of the micro-rotations. Although the CORDIC rotation can be executed in parallel according to [32], the method for decomposition of each positional binary weight produces many extra stages of micro-rotation, especially when the bit-width of input angle increases. A more efficient recoding scheme has been proposed in [33] for the reduction of number of micro-rotations to be employed in parallel CORDIC rotations.

Redundant-Number-Based CORDIC Implementation

Addition/subtraction operations are faster in the redundant number system, since unlike the binary system, it does not involve carry propagation. The use of redundant number system is therefore another way to speed up the CORDIC iterations. A CORDIC implementation based on the redundant number system called as redundant CORDIC was proposed by Ercegovac and Lang and applied to matrix triangularization and singular value decomposition [34]. Rotation mode redundant CORDIC has been found to result in fast implementation of sinusoidal function generation, unitary matrix transformation, angle calculation and rotation [34]–[38].

Pipelined CORDIC Architecture

Since the CORDIC iterations are identical, it is very much convenient to map them into pipelined architectures. The main emphasis in efficient pipelined implementation lies with the minimization of the critical path. The earliest pipelined architecture that we find was suggested by Deprettere, Dewilde and Udo in 1984 [14]. Pipelined CORDIC circuits have been used thereafter for high-throughput implementation of sinusoidal wave generation, fixed and adaptive filters, discrete orthogonal transforms and other signal processing applications [40]–[44]. A generic architecture of pipelined CORDIC circuit is shown in Fig. 7. It consists of stages of CORDIC units where each of the pipelined stages consists of a basic CORDIC engine of the kind shown in Fig. 2. Since the number of shifts to be performed by the shifters at different stages is fixed (shift-operation through -bit positions is performed at the th stage) in case of pipelined CORDIC the shift operations could be hardwired with adders; and therefore shifters are eliminated in the pipelined implementation. The critical-path of pipelined CORDIC thus amounts to the time required by the add/subtract operations in each of the stages.

IV. APPLICATIONS OF CORDIC

CORDIC technique is basically applied for rotation of a vector in circular, hyperbolic or linear coordinate systems, which in turn could also be used for generation of sinusoidal waveform, multiplication and division operations, and evaluation of angle of rotation, trigonometric functions, logarithms, exponentials and squareroot [6], [64], [65]. Table IV shows some elementary functions and operations that can be directly implemented by CORDIC. The table also indicates whether the coordinate system is circular (CC), linear (LC), or hyperbolic (HC), and whether the CORDIC operates in rotation mode (RM) or vectoring mode (VM), the initialization of the CORDIC and the necessary pre- or postprocessing step to perform the operation. The scale factors are, however, obviated in Table IV for simplicity of presentation. In this Section, we discuss how CORDIC is used for some basic matrix problems like QR decomposition and singular-value decomposition. Moreover, we make a brief presentation on the applications of CORDIC to signal and image processing, digital communication, robotics and 3-D graphics.

Matrix Computation

Singular Value Decomposition and Eigenvalue Estimation

Signal Processing and Image Processing Applications

V. CONCLUSION

The beauty of CORDIC is its potential for unified solution for a large set of computational tasks involving the evaluation of trigonometric and transcendental functions, calculation of multiplication, division, square-root and logarithm, solution of linear systems, QR-decomposition, and SVD, etc. Moreover, CORDIC is implemented by a simple hardware through repeated shift-add operations. These features of CORDIC has made it an attractive choice for a wide variety of applications. In the last fifty years, several algorithms and architectures have been developed to speed up the CORDIC by reducing its iteration counts and through its pipelined implementation. Moreover, its applications in several diverse areas including signal processing, image processing, communication, robotics and graphics apart from general scientific and technical computations have been explored. Latency of computation, however, continues to be the major drawback of the CORDIC algorithm, since we do not have efficient algorithms for its parallel implementation. But, CORDIC on the other hand is inherently suitable for pipelined designs, due to its iterative behavior, and small cycle time compared with the conventional arithmetic. For high-throughput applications, efficient pipelined-architectures with multiple-CORDIC units could be developed to take the advantage of pipelineability of CORDIC, because the digital hardware is getting cheaper along with the progressive device-scaling. Research on fast implementation of shift-

accumulation operation, exploration of new number systems for CORDIC, optimization of CORDIC for constant rotation have scope for further reduction of its latency. Another way to use CORDIC efficiently, is to transform the computational algorithm into independent segments, and to implement the individual segments by different CORDIC processors. With enhancement of its throughput and reduction of latency, it is expected that CORDIC would be useful for many high-speed and real-time applications. The area-delay-accuracy trade-off for different advanced algorithms may be investigated in detail and compared with in future work.

REFERENCES

[1] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Computers, vol. EC-8, pp. 330–334, Sept. 1959. [2] J. E. Volder, "The birth of CORDIC," J. VLSI Signal Process., vol. 25, pp. 101–105, 2000. [3] J. S. Walther, "A unified algorithm for elementary functions," in Proc. 38th Spring Joint Computer Conf., Atlantic City, NJ, 1971, pp. 379–385.

[2] J. S. Walther, "The story of unified CORDIC," J. VLSI Signal Process., vol. 25, no. 2, pp. 107–112, June 2000.

[3] D. S. Cochran, "Algorithms and accuracy in the HP-35," HewlettPackard J., pp. 1–11, Jun. 1972. [6] J.-M. Muller, Elementary Functions: Algorithms and Implementation. Boston, MA: Birkhauser Boston, 2006.

[4] S.-F. Hsiao and J.-M. Delosme, "Householder CORDIC algorithms," IEEE Trans. Computers, vol. 44, no. 8, pp. 990–1001, Aug. 1995.

[5] E. Antelo, J. Villalba, and E. L. Zapata, "A low-latency pipelined 2D and 3D CORDIC processors," IEEE Trans. Computers, vol. 57, no. 3, pp. 404–417, Mar. 2008.

[6] D. Timmermann, H. Hahn, and B. J. Hosticka, "Low latency time CORDIC algorithms," IEEE Trans. Computers, vol. 41, no. 8, pp. 1010–1015, Aug. 1992.

[7] S. Wang, V. Piuri, and J. E. E. Swartzlander, "Hybrid CORDIC algorithms," IEEE Trans. Computers, vol. 46, no. 11, pp. 1202–1207, Nov. 1997.