A High-Speed CORDIC Algorithm and Architecture for DSP Applications [*]

Martin Kuhlmann, Keshab K. Parhi
Dept. of Electrical and Computer Engineering
University of Minnesota, Minneapolis MN 55455, USA
Email: {kuhlmann, parhi}@ece.umn.edu

Abstract - This paper presents a novel CORDIC algorithm and architecture for the rotation and vectoring mode in circular coordinate systems in which the directions of all micro-rotations are precomputed while maintaining a constant scale factor. Thus, an examination of the sign of the angle or y-remainder after each iteration is no longer required. By using Most-Significant Digit (MSD) first adder/multiplier, the critical path of the entire CORDIC architecture only requires $(1.5n + 2)$ and $(1.5n + 10)$ full-adders (n corresponds to the word-length of the inputs) for rotation and vectoring modes, respectively. This is a speed improvement of about 30% compared to the previously fastest reported shared rotation and vectoring mode implementations. Additionally, there is a higher degree of freedom in choosing the pipeline cutsets due to the novel independence of iteration i and i-1 in the CORDIC rotation. Optional pipelining can lead for example to an on-line delay of three clock-cycles where every clock cycles corresponds to a delay of twelve full-adders.

1 Introduction

CORDIC (COordinate Rotation DIgital Computer) [1, 2] is an iterative algorithm for the calculation of the rotation of a two-dimensional vector, in linear, circular or hyperbolic coordinate systems, using only add and shift operations. Its current applications are in the field of digital signal processing, image processing, filtering, matrix algebra, etc [3].

The CORDIC algorithm consists of two operating modes, the rotation mode and the vectoring mode, respectively. In the rotation mode, a vector (x,y) is rotated by an angle β to obtain the new vector (x^*, y^*). In every micro-rotation i, fixed angles of the value $\arctan(2^{-i})$ which are stored in a ROM are subtracted or added from/to the angle remainder θ_i, so that the angle remainder approaches zero. In the vectoring mode, the length R and the angle towards the x-axis α of a vector (x,y) are computed (see Fig. 1). For this purpose, the vector is rotated towards the x-axis so that the y-component

[*]This work was supported by the Defense Advanced Research Projects Agency under contract number DA/DABT63-96-C-0050.

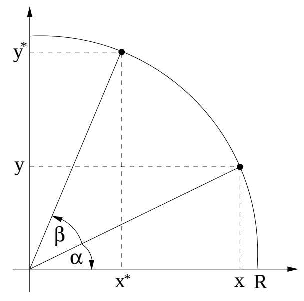


Figure 1: The rotation and vectoring mode of the CORDIC algorithm

approaches zero. The sum of all angle rotations is equal to the value of α, while the value of the x-component corresponds to the length R of the vector (x, y). Eq. (1-3) show the mathematical equations for the CORDIC rotations,

$$x_{i+1} = x_i - m \cdot \sigma_i \cdot 2^{-i} \cdot y_i, \tag{1}$$

$$y_{i+1} = y_i + \sigma_i \cdot 2^{-i} \cdot x_i, \tag{2}$$

$$z_{i+1} = z_i - \frac{1}{m} \cdot \sigma_i \cdot \arctan(\sqrt{m} 2^{-i}), \tag{3}$$

where σ_i is either $+1$ or -1 depending on the sign of the z_i or y_i component, depending on rotation or vectoring mode, respectively and m steers the choice of linear, circular or hyperbolic coordinate systems. The required micro-rotations are not perfect rotations, they increase the length of the vector. Nevertheless, assuming consecutive rotations in positive and/or negative directions, the scale factor can be precomputed according to

$$K = \prod_{i=0}^{n} k_i = \prod_{i=0}^{n} (1 + \sigma_i^2 \cdot 2^{-2i})^{1/2}. \tag{4}$$

The computation of the scale factor can be truncated after $n/2 + 1$ iterations because the multiplicands in the last $n/2$ iterations are 1 due to the finite word-length and do not affect the final value of K.

To speed up the computation, various implementations have been proposed. In [4] a double rotation method is introduced which guarantees a constant scale factor while only the first three MSDs are examined (rotation mode only). This method is extended to the vectoring mode in [5]. To reduce the latency of the CORDIC operation, an on-line algorithm has been proposed in [6] . However, this results in a variable scale factor. This drawback is removed in [7]. Yet, in every iteration a sign check is required. The fastest implementation requires a delay of $3n$ full-adder delays (not including the scaling operation). Other implementations like [8] remove the extra rotations by a branching mechanism in case that the sign of the remainder cannot be determined. This effectively doubles the required implementation area. In [9], the signs of all micro-rotations are still computed serially. However, a speed-up of the sampling rate is achieved by separating the computation of the sign and the magnitude of every z_i or y_i remainder. The sign of every remainder is computed by a pipelined Carry-Ripple Adders (CRA) leading to an initial latency of n full-adders before the first CORDIC rotation can be performed. Nevertheless, after this initial latency, the following signs can be obtained with a delay of only one full-adder. This leads to an overall latency of $3n$ and $3.5n$ full-adders delays for rotation and vectoring mode, respectively. In [10], the rotation direction for the last $2n/3$ rotations have been determined in parallel, while for the first $n/3$ rotations a standard approach has been used. However, this scheme only works for the rotation mode. Alternative to the CORDIC implementations with constant scale factor, other implementations use a minimally redundant radix-4 number representation [11, 12, 13]. By using this number system, the number of iterations can be reduced by a factor of two. However, the scale factor becomes variable and has to be computed every time, due to the absence of consecutive rotations. This leads to an increase of complexity.

This paper is organized as follows. Section 2 presents the theoretical background for the novel CORDIC algorithm for rotation and vectoring mode in circular coordinates. Section 3 presents the new architecture of the algorithm which leads to the fastest implementation. Section 4 concludes the paper.

2 The Novel CORDIC Algorithm

2.1 The Rotation Mode

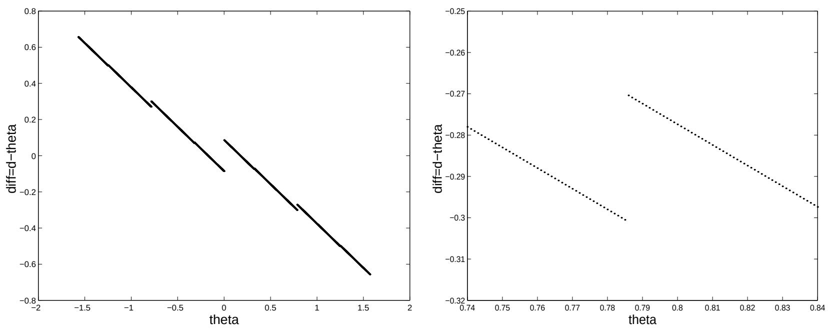
In [10, 14], a relationship between the rotation angle θ and the directions of the micro-rotations has been already discovered. However, in [14], the correlation is only mentioned for $m = 0$ (CORDIC in linear coordinate systems), while in [10] a correlation between the last $2n/3$ rotation directions and the angle remainder is described, where n represents the word-length.

There is, however, a global correlation between the binary representation of θ and the directions of the micro-rotations. The representation of the directions of all micro-rotations (abbr. as d) can be written as

$$d = \sum_{i=0}^{n-1} a_i \cdot 2^{-i} = a_0.a_1a_2a_3...a_{n-1}, \tag{5}$$

where $a_i \in \{\bar{1}, 1\}$. A *one* always represents a positive micro-rotation (the angle remainder is positive, so the value of $\arctan(2^{-i})$ is subtracted), a $\bar{1}$ always represents a negative micro-rotation (the angle remainder is negative, hence the value of $\arctan(2^{-i})$ is added). This representation is called Offset-Binary Coding (OBC) [15, 16].

In order to convert this number into a binary number, all $\bar{1}$ are regarded as 0. All digits except the Most Significant Digit (MSD) are shifted one position towards left. The MSD is also shifted one position to the left. However, its value is the inverse of a_0. If a_0 is 1 then the MSB of the binary representation is 0, if a_0 is 0 (which corresponds to $\bar{1}$) then the MSB is 1. By comparing this constructed binary representation of d with the input angle θ a correlation can be obtained (Fig. 2a). The figure shows the correlation from angles in the range of $-\pi/2$ to $\pi/2$. However, Fig. 2 (b) reveals that



a) b)

Figure 2: The correlation for the rotation mode between the rotation directions and the given rotation angle θ. The figures show the range from $\theta = -\pi/2$ to $\theta = \pi/2$ (a), the zoom-in from $\theta = 0.74$ to $\theta = 0.84$ (b)

there are break points in the graph. Nevertheless, these break points and the magnitude of the discontinuities can be exactly calculated.

The slope of the difference between the sought word d and the angle θ is exactly 0.5 after removing all discontinuities. Hence, it does not require a multiplication but only an add operation. By using these results following correlation can be obtained for the computation of the directions of the micro-rotations

$$d = \theta - 0.5\theta + b \pm \epsilon, \tag{6}$$

where b represents the positive and negative value of d (depending on a positive or negative θ) at the point $\theta = 0$ and ϵ is the sum of all discontinuities in the graph from 0 to θ. The computation of the first three terms of (6) is trivial. However, to compute ϵ it requires some more effort.

2.1.1 Characteristics of the Discontinuities

The breakpoints of the graph shown in Fig. 2(a) can be calculated analytically. These break points are caused by the redundant representation of the rotation directions. The breakpoints bp_i are located at

$$bp_i = m_0 \arctan(2^0) + m_1 \arctan(2^{-1}) + ... + m_i \arctan(2^{-i}), \tag{7}$$

where m_i is either $1, 0$ or -1. However, the larger i, the smaller the jump at the break point bp_i. Note that if $m_i = 0$, all $m_{l>i} = 0$. The largest jump (besides that one at 0) is located at $bp = \arctan(2^0) = \pi/4$ (see Fig. 2(b)).

Table 1 shows the decimal and binary value of the largest discontinuity and the corresponding directions of the micro-rotations, respectively. Obviously, a small change in θ causes a large change in d. By taking the difference between the two values in the third column, δ_d can be computed

$$\delta_d = 0.00000111110101001111011011111 = 0.03059360291809. \tag{8}$$

Table 2 shows all break points and their corresponding values of δ_d for a precision of 16 bits. Every additional set of break points which effectively doubles the entire number of break points, leads to an additional precision of three bits of ϵ.

Table 1: The largest discontinuity in rotation mode

θ_{dec}	θ_{binary}	directions of the micro-rotations
0.7853981628	0.11001001000011111101101010100111	10111110000010101100000010010000
0.7853981634	0.11001001000011111101101010101000	11000001111101010011111101101111

Table 2: The break points of the rotation mode and their corresponding values of δ_d

Break point	δ_d	Break point	δ_d
0.0148357067	8.71438533067e-06	0.7853981634	0.030593602918
0.0767718912	5.52195124328e-04	0.8172933047	8.71438533067e-06
0.1387080758	8.71438533067e-06	0.8797121147	6.95707276463e-05
0.2011268858	6.95707276463e-05	0.9421309247	8.71438533067e-06
0.2635456958	8.71438533067e-06	1.0040671093	5.52195124328e-04
0.3217505543	0.004283527843	1.0660032938	8.71438533067e-06
0.3799554129	8.71438533067e-06	1.1284221038	6.95707276463e-05
0.4423742229	6.95707276463e-05	1.1908409138	8.71438533067e-06
0.5047930329	8.71438533067e-06	1.2490457724	0.004283527843
0.5667292175	5.52195124328e-04	1.3072506310	8.71438533067e-06
0.6286654020	8.71438533067e-06	1.3696694410	6.95707276463e-05
0.6910842120	6.95707276463e-05	1.4320882510	8.71438533067e-06
0.7535030220	8.71438533067e-06	1.4940244355	5.52195124328e-04

2.2 The Vectoring Mode

In [10, 14], no correlation between the rotation angle θ and the directions of the micro-rotations for the vectoring mode is reported. Similar to the rotation mode in circular coordinate systems, there is also a global correlation between the binary representation of y and the directions of the micro-rotations. The representation of the directions of all micro-rotations is chosen to be the same as for the rotation mode (see (5)).

The correlation for the vectoring mode between the x and y coordinates and the rotation directions are not as obvious as in the rotation mode. An angle of $\pi/4$ can result from equal x and y coordinates, however, by increasing/decreasing the length of the vector by the factor k, the angle does not change, but $k \cdot x$ and $k \cdot y$ might look totally different from x and y. Nevertheless, both representation lead to the same rotation directions. By plotting the binary representation of d over the binary representation of x and y, Fig. 3 can be obtained. By examining discrete value of x between 1 and 2, Fig 3 (c) and (d) can be obtained. Since we limited the range of the angle θ to $\arctan(2) = 63.4$ degree, only y-coordinates with exponents smaller or equal to x are used (see Fig. 3 (c)). By limiting the maximal rotation angle to 45 degrees, (y is always smaller than x), Figure 3(d) is obtained. The restriction of 45 degrees leads to a smaller ROM-size. Additionally, the entire sum of all discontinuities is smaller than 2^{-7}, which results in an overall speed-up. However, it also requires a full-length comparison of the x and y components at the beginning.

The choice of limiting x between 1 and 2 is no restriction of the angle θ between the vector (x, y) and the x-axis. The x and y operands just have to be shifted according to the position of the leading 1 of x, and the resulting length of the vector R has to be shifted back.

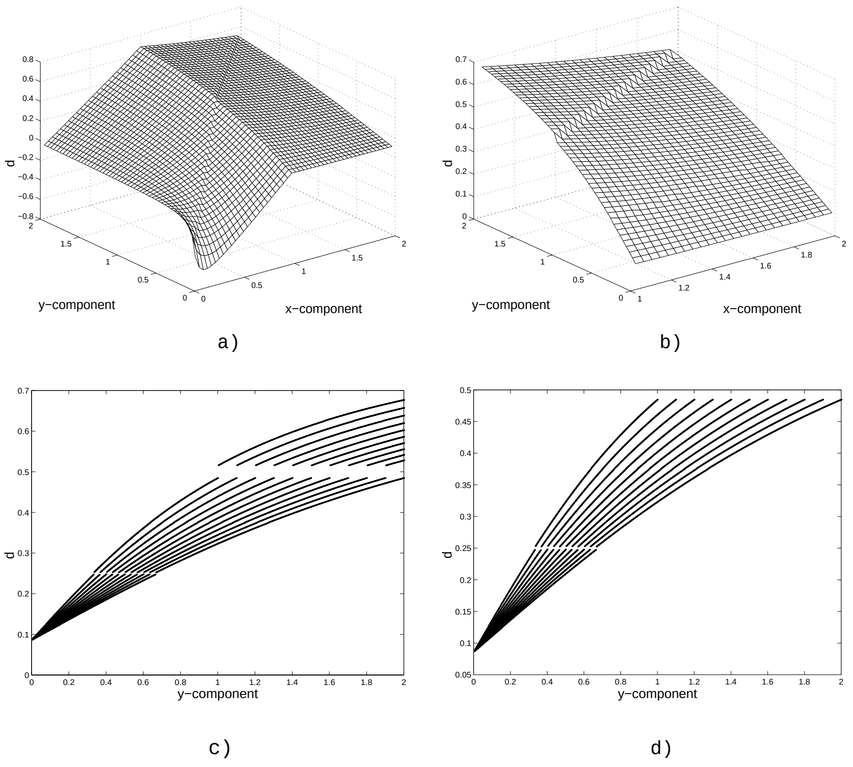


Figure 3: The correlation for the vectoring mode between the rotation directions and the given rotation angle θ. The figures show the 3-d plot (a) for x and y in the range from 0 to 2, (b) the zoom-in of the 3-d plot for x in the range from 1 to 2, (c) the 2-d plot for discrete values of x (most left 1.0, most right 2.0, step-size 0.1) and (d) the same as (c), but limiting the maximal y-component to be smaller or equal to x (45 degree line).

In Fig. 3(c) it is obvious that every graph of the function $f_x(y)$ (in Fig. 3(c) there are entirely shown ten different graphs) is only a stretch of $f_{x=1}(y)$ by x in the y-direction. This leads to following relationship

$$y = x \cdot f_{x=1}(d), \tag{9}$$

where $f_{x=1}(d)$ is the function of $x = 1$. However, this function can not be obtained trivially. Nevertheless, we can sub-divide the graph into 2^{n_0} segments and interpolate the subgraphs by linear graphs. Thus, the latter equation can be reformulated into

$$y = x \cdot (m_i \cdot d + n_i), \tag{10}$$

where m_i and n_i are the slope and the y-intersect of the corresponding segment. Eq. (10) can be easily solved for d which leads to

$$d = \left(\frac{y}{x} - n_i\right) \cdot \frac{1}{m_i} = \frac{y}{x} \cdot m_i' + n_i', \tag{11}$$

where m_i' and n_i' are the inverse of m_i and negative quotient of n_i by m_i, respectively. However, there is a division required which is as expensive as the entire CORDIC operation. Nevertheless, instead of performing a division, y and x can be scaled by a constant k, so that $x \cdot k$ lies in the range $[1, 1+\Delta)$. This leads to

$$\begin{aligned} d &= \frac{y}{x} \cdot m_i' + n_i' = \frac{y \cdot k}{x \cdot k} \cdot m_i' + n_i' = \frac{y \cdot k}{1+\delta} \cdot m_i' + n_i' && (12) \\ &= (y \cdot k)(1-\delta) \cdot m_i' + (1-\delta) \cdot n_i'. && (13) \end{aligned}$$

Simulations have shown that the last transformation in (13) can be performed if $0 < \delta < 1/32$ and two ROMs (for m and n each) are used to store the coefficients m and n. The ROMs are addressed by y and by the seventh bit of $x \cdot k$ which corresponds to δ_7. The ROMs consists of 32 entries of 16 bits.

3 The Novel CORDIC Architectures

3.1 The Rotation Mode

The pre-processing part of the new architecture (see Fig. 4) consists of a ROM of 26 entries in which the breakpoints bp_i and the corresponding errors ϵ_i and ϵ_{i+1} are stored, respectively (see Table 2). The ROM is accessed by the five MSB bits of θ. In case $\theta < 0$, it is sufficient to invert all bits and access the ROM with the 5 MSB bits, the additional one can be added while performing $\theta \pm bp_i$. All values have a word-length of n. The corresponding value of b ($b = \pm 0.0863277670$) is selected according to the sign of the angle θ and added to $0.5 \cdot \theta$, which is done by a redundant adder which has a word-length independent critical path of one full-adder (maximally redundant radix-4 adder). The maximal magnitude of ϵ is 0.03692625783 (this corresponds to the sum of all discontinuities) and is less than 2^{-4}. Hence, the first two CORDIC rotations (equal to four full-adder delays) can be done without knowing the value of ϵ. In the meantime, a fast tree-adder [17] can be used to subtract/add the breakpoint bp_i from/to θ. This adder requires a delay of four full-adders (tree-adder). Depending on the sign of the result either ϵ_i or ϵ_{i-1} is selected and added/subtracted (depending on the sign of θ) to/from the sum of θ and b. An on-the-fly converter converts the redundant result into binary format so that the directions of the micro-rotations can be obtained. Due to the redundant characteristics of the fixed angle arguments $\arctan(2^{-i})$, where

$$\arctan(2^{-i}) < \sum_{k=i+1}^{i+l} \arctan(2^{-k}), \tag{14}$$

the rotations directions of iteration i is not influenced by the rotation directions of iteration $i + l$. Hence, it is not possible that there are more than

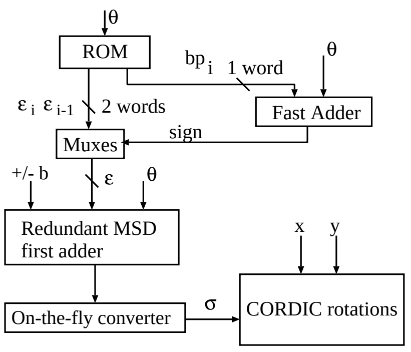


Figure 4: The novel architecture for the proposed CORDIC algorithm (rotation mode)

$l-1$ consecutive rotations in the same direction. In case that there are $l-1$ consecutive rotations in the same direction, the l^{th} iteration has to be rotated into the opposite direction. This happens if the angle remainder $z_i \approx 0$. Table 3 shows the number of consecutive rotation in the same direction for different values of i. This limitation leads to a reduction in the complexity of the on-the-fly converters and its most significant bits can already be used to start the CORDIC operation. Note that in every iteration two digits are computed and can be converted into the corresponding binary representation (equals four bits). Hence, the entire delay of the rotation mode is equivalent to $1.5n + 2$ full-adder delays.

Table 3: The maximal number of consecutive rotation in the same direction

i	angle θ_i	$l-1$	i	angle θ_i	$l-1$	i	angle θ_i	$l-1$
0	45	3	2	14.04	6	4	3.58	10
1	26.57	5	3	7.13	8	5	1.79	12

3.2 The Vectoring Mode

In the vectoring mode, the computation of the rotation directions requires the scaling of the x and y operands. The scaled $k \cdot y$ (see Fig. 5) operand selects according to the four and six most significant bits the discontinuity ϵ and the coefficients m and n, respectively. Note that one bit from the scaled operand $k \cdot x$ is also required to determine the coefficients. Similar to the rotation mode, the discontinuities must be added to obtain the accurate directions of the micro-rotations. The breakpoints bp_i are stored for the normalized case of $x = 1$. Note that the breakpoints of the vectoring mode correspond to the tangent of the breakpoints of the rotation mode. However, the value of the discontinuities remains the same. After scaling, $k \cdot x$ might not be equal of 1. Nevertheless, instead of computing the correct quotient y/x and subtract the breakpoint from it, it is sufficient to examine $y \cdot k - (1+\delta)bp$. Note that δ is small and by using Booth encoding the entire number of partial sums can be reduced to eight. A modified tree adder only computes the sign of the output (the actual value is not required). The sign selects the

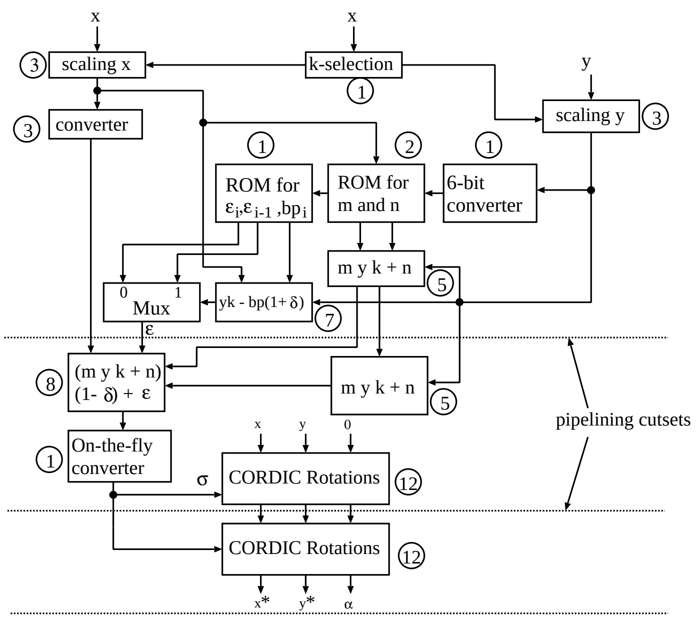


Figure 5: The novel architecture for the proposed CORDIC algorithm (vectoring mode)

corresponding discontinuity ϵ_i or ϵ_{i-1}. In the meantime, the output digits of $m \cdot y \cdot k + n$ are computed most significant digit first. Using these digits, the result of $(m \cdot y \cdot k + n) \cdot (1 - \delta)$ is obtained which is immediately converted into the equivalent binary representation. The first direction of the micro-rotation is always positive, hence, even before the converter obtains its first output, the CORDIC rotation starts its computation. Due to the limited consecutive ones/zeros of the converter (see Table 3), the remaining rotation can be executed without any delay. This results in an entire delay of $1.5n+10$ full-adder delays, which corresponds to a speed-up of 30%. The pipelined architecture shown in Fig. 5 has an on-line delay of three clock cycles, where every clock cycles consists of approximately 12 full-adder delay. The circled numbers give an approximate delay of the elements used in the architecture. Note, that the critical path is not equivalent to the sum of all numbers since there are MSD first adders and multipliers. This corresponds to a delay of 36 full-adder delays and hence is 25% faster than any other reported CORDIC implementation. To obtain the highest sample rate, the architecture can be pipelined after each full-adder.

4 Conclusion

This paper has presented for the first time a CORDIC algorithm to compute the directions of the required micro-rotations on-line and before the actual CORDIC computation starts. This is obtained by using a linear correlation between the rotation angle θ and the corresponding direction of all micro-rotations for the rotation mode and by a linear interpolation for the vectoring mode. Hence, neither extra nor double rotations nor a variable scale factor are required. Optional pipelining can lead to an on-line delay of three clock cycles, where one clock cycles corresponds to the delay of twelve full-adders. This corresponds to a speed-up of about 25% over all previously reported CORDIC implementations. The implementation is suitable for word-length up to 19 bits with a reasonable ROM size.

References

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Elect. Comput.*, vol. EC, pp. 330–334, Aug. 1959.

[2] J. S. Walther, "A unified algorithm for elementary functions," *Proc. Spring. Joint Comput. Conf.*, vol. 38, pp. 379–385, 1971.

[3] Y. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Processing Magazine*, pp. 16–35, Jul. 1992.

[4] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Trans. on Computers*, vol. 40, pp. 989–995, Sep. 1991.

[5] J. Lee and T. Lang, "Constant-factor redundant CORDIC for angle calculation and rotation," *IEEE Trans. on Computers*, vol. 41, pp. 1016–1025, Aug. 1992.

[6] H. Lin and A. Sips, "On-Line CORDIC Algorithms," *IEEE Trans. on Computers*, vol. 39, pp. 1038–1052, Aug. 1990.

[7] R. Hamill, J. McCanny, and R. Walke, "On-Line CORDIC Algorithm and VLSI Architecture for Implementing QR-Array processors," *Journal of VLSI Signal Processing*, p. to be published, 1999.

[8] J. Duprat and J.-M. Muller, "The CORDIC Algorithm: New results for Fast VLSI Implementation," *IEEE Trans. on Computers*, vol. 42, pp. 168–178, Feb. 1993.

[9] H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundnat implementation without correcting iterations," *IEEE Trans. on Computers*, vol. 45, pp. 307–318, Mar. 1996.

[10] S. Wang, V. Piuri, and E. Swartzlander, "Hybrid CORDIC Algorithms," *IEEE Trans. on Computers*, vol. 46, pp. 1202–1207, Nov. 1997.

[11] C. Li and S. Chen, "A radix-4 redundnat CORDIC algorithm with fast on-line variable scale factor compensation," in *International Symposium on Circuits and Systems*, (Hong Kong), pp. 639–642, Jun. 1997.

[12] R.R.Osorio, E. Antelo, J. Bruguera, J. Villalba, and E. Zapata, "Digit on-line large radix CORDIC rotator," in *International Conference on Applications-Specific Array Processors*, (Strasbourg, France), pp. 247–257, Jul. 1995.

[13] J. Villalba, J. Hidalgo, E. Zapata, E. Antelo, and J. Bruguera, "CORDIC architectures with parallel compensation of the scale factor," in *International Conference on Applications-Specific Array Processors*, (Strabourg, France), pp. 258–269, Jul. 1995.

[14] D. Timmermann, H. Hahn, and B. Hosticka, "Low latency time CORDIC algorithms," *IEEE Trans. on Computers*, vol. 41, pp. 1010–1015, Aug. 1992.

[15] N. Demassieux, F. Jutand, and P. Pirsch, *VLSI implementation for image communications*, ch. 7. New York: Elsevier Science, 8th ed., 1993.

[16] K. K. Parhi, *VLSI Digital Signal Processing, Design and Implementation*, ch. 7. New York: Wiley & Sons, 1999.

[17] K. K. Parhi, "Low-Energy CSMT carry Generation & Binary Addition," *IEEE Trans. on VLSI Systems*, vol. 7, Dec. 1999.