A Novel CORDIC Rotation Method for Generalized Coordinate Systems *

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

CORDIC (COordinate Rotation DIgital Computer) 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. This paper presents a novel algorithm and architecture for the rotation mode in circular and hyperbolic 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 after each iteration is no longer required. By using redundant adder, the critical path (without scaling and conversion) of the entire CORDIC architecture only requires $(1.5n + 2)$ full-adders (n corresponds to the word-length of the inputs) for rotation mode. This is a speed improvement of about 20% compared to the previously fastest reported rotation mode implementations. Additionally, there is a higher degree of freedom in choosing the pipeline cutsets due to the novel feature of independence of the iterations i and i-1 in the CORDIC rotation. Optional pipelining can lead for example in the rotation mode to an on-line delay of three clock-cycles including scaling and conversion, where every clock cycle 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 θ 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

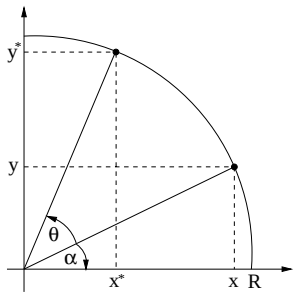

Figure 1. The rotation and vectoring mode of the CORDIC algorithm

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 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). The mathematical relations for the CORDIC rotations are shown in (1-3),

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

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

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

where σ_i is either $+1$ or -1 depending on the sign of the z_i (angle remainder) or y_i component, depending of rotation or vectoring mode, and m steers the choice of rectangular $(m = 0)$, circular $(m = 1)$ or hyperbolic $(m = -1)$ coordinate systems. The required micro-rotations are not perfect rotations, they increase the length of the vector. In order to maintain a constant vector length, the obtained results have to be scaled by a scale factor K. Nevertheless, assuming consecutive rotations in positive and/or negative directions,

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

the scale factor can be precomputed according to

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

The computation of the scale factor can be truncated after $(n/2 + 1)$ iterations because the multiplicands in the last $(n/2 - 1)$ 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, [6] proposed an algorithm using on-line arithmetic. However, this results in a variable scale factor. This drawback is removed in [7]. However, in every iteration a sign check is required resulting in an additional delay. The fastest implementation requires a delay of $3n$ full-adder delays (not including the scale operation and the conversion). 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. Nevertheless, the computation time still requires a delay of $3n$ full-adder delays. In [9], the signs of all micro-rotations are 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 Adder (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 are determined in parallel, while for the first $n/3$ rotations a standard approach is 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 in complexity.

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

2 The Novel CORDIC Algorithm

2.1 The Rotation Mode for circular coordinate systems

In [10, 14], a relationship between the rotation angle θ and the directions of the micro-rotations was already presented. However, in [14], the correlation is mentioned for $m = 0$ and $m = 1$. As in [14] indicated, the conversion can be accomplished through a simple recoding in linear coordinate, but requires more complicated treatment in circular and hyperbolic coordinate systems. 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 [15]. The representation of the directions of all micro-rotations d can be written as

$$d = \sum_{i=1}^{n} d_i \cdot 2^{-i} = d_0.d_1 d_2 d_3 ... d_{n-1}, \quad (5)$$

where $d_i \in \{\bar{1}, 1\}$. 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) [16].

To find a correlation between the binary representation of θ and the directions of the micro-rotations, this number is converted into a binary number in the following manner. All $\bar{1}$ are regarded as 0. However, the MSD digit is the inverse of d_0. If $d_0 = 1$ then the MSB of the binary representation is 0, if $d_0 = 0$ (which corresponds to $\bar{1}$ then the MSB of d is one (this is due to the 2's complement representation of the angle θ). By comparing this constructed binary representation of d with the input angle θ a correlation can be obtained (Fig. 2). The figure shows the correlation from angles in the range of $-\pi/2$ to $\pi/2$. However, Fig. 3 reveals that 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 (see Fig. 4,5). Hence, it does not require a multiplication but only one shift-add operation. By using these results the following correlation can be obtained for the computation of the directions of the micro-rotations

$$d = \theta - 0.5\theta + b \pm \epsilon = 0.5\theta + b \pm \epsilon, \quad (6)$$

where b represents the positive and negative value of d (depending on a positive or negative value of 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 two terms of (6) is

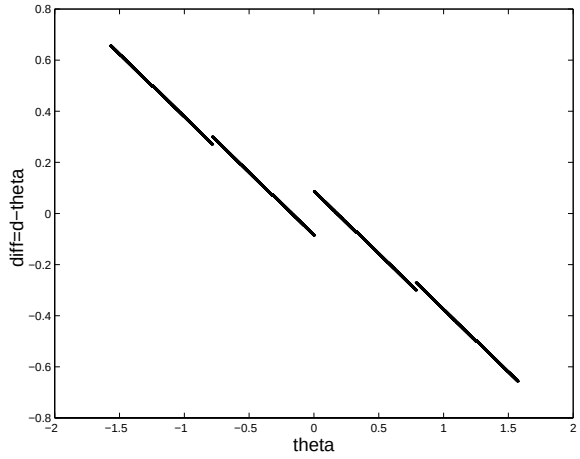

Figure 2. The correlation for the rotation mode between the rotation directions and the given rotation angle θ. The figures show the range from $\theta = -\pi/2$ to $\theta = \pi/2$

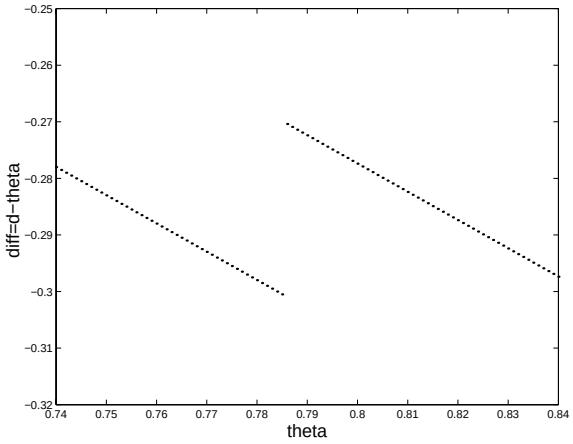

Figure 3. The correlation for the rotation mode between the rotation directions and the given rotation angle θ. The figures show the range from $\theta = 0.74$ to $\theta = 0.84$ (b)

trivial. However, to compute ϵ it requires some more effort. Note, that the pre-computation for the rotation directions d eliminates the z-data-path, since in the rotation mode, the angle computation is only performed to determine the rotation directions. Hence, the area of the implementation is reduced.

2.1.1 Characteristics of the Discontinuities

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

$$bp_{rm,i} = g_0 \mathrm{atan}(2^0) + \mathrm{g_1 atan}(2^{-1}) + ... + \mathrm{g_i atan}(2^{-\mathrm{i}}), \quad (7)$$

where g_i are either $1, 0$ or -1. However, the larger the value of i, the smaller the jump at the break point bp_i. Note that if $g_i = 0$, all $g_{l>i} = 0$ (this is due to the consecutive rotations). The largest jump (besides the one at 0) is located at $bp = \arctan(2^0) = \pi/4$ (see Fig. 3).

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

$$\delta_d = 0.00000111110101001111101101111_2 \quad (8)$$
$$= 0.03059360291809_{10}. \quad (9)$$

Table 2 shows all 26 possible break points and their corresponding values of δ_{d_i} 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.

Table 1. The largest discontinuity in rotation mode

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

High Precision By using a mantissa of 54 bits (which corresponds to the floating point representation), the ROM for storing all discontinuities would require 2^{16} entries (the value of ϵ_{17} is smaller than 2^{-55}). This is rather impractical since the required area to implement the ROM will exceed by far the area for the CORDIC implementation.

Nevertheless, this bottleneck can be eliminated by realizing that the following relationship between the $\arctan(2^{-i})$ and its binary representation can be obtained for $i \geq 9$

$$\arctan(2^{-i}) = 2^{-i} - \frac{1}{3}2^{-3i}. \quad (10)$$

This leads approximately to linear distribution of the discontinuities. By using 2^{-i} instead of $\arctan(2^{-i})$ for the selection of the breakpoint, the error $\Delta_{epsilon_i}$ is approximately

$$\epsilon_{3i} \approx \epsilon_i \cdot 2^{-3 \cdot 3i}, \quad (11)$$

which is smaller than 2^{-54} for $i \geq 9$. Hence, the bits 9 to 17 of the absolute value of the difference $bp_{diff} = |\theta - bp_i|$

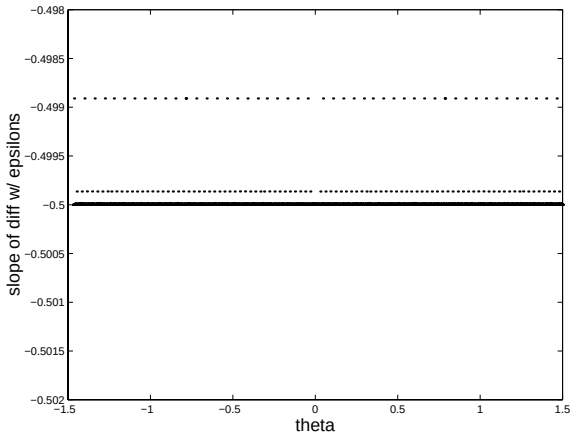

Figure 4. The slope of the rotation direction as a function of θ. The discontinuities in Fig. 2 are reflected by the discontinuities in the slope. The figures show the range from $\theta = -\pi/4$ to $\theta = \pi/4$

can be used to determine the remaining error ϵ_r. Note that the first eight bits of bp_{diff} are zero.

Hence, a ROM with only 2^8 entries is required, each storing the break point bp_i and the value of the discontinuity ϵ_{i-1} and ϵ_i, respectively. The first eight bits of the binary representation of θ (there are no two break points that have the same eight leading bits) are used as the address. In order to determine which ϵ has to be used, a subtraction of θ and the breakpoint bp_i has to be performed. The remaining discontinuity ϵ_r is computed according to

$$\epsilon_r = \sum_{j=9}^{17} s_j \cdot \epsilon_j, \tag{12}$$

where the coefficients s_j are computed according to

$$s_j = \sum_{l=9}^{j-1} (bp_l \cdot 2^{17-l}) + bp_{diff,j}, \tag{13}$$

where bp_l and $bp_{diff,j}$ represent the l^{th} and the j^{th} bit of bp_{diff}, respectively.

By realizing, that the discontinuities ϵ_i have the following relationship for $i \geq 9$

$$\epsilon_i = 8 \cdot \epsilon_{i+1}, \tag{14}$$

the computation of the remaining error ϵ_r can be optimized. An efficient way to compute the coefficients s_j is shown in Fig. 5. In order to obtain the digits of s_j, two rows of full-adders and an additional Carry-Propagate Adder (CPA) are required. Nevertheless, due to (14), s can be interpreted as a

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

Break point	δ_d	Break point	δ_d
0.0148357	8.71438533e-06	0.7853981	0.030593602918
0.0767718	5.52195124e-04	0.8172933	8.71438533e-06
0.1387080	8.71438533e-06	0.8797121	6.95707276e-05
0.2011268	6.95707276e-05	0.9421309	8.71438533e-06
0.2635456	8.71438533e-06	1.0040671	5.52195124e-04
0.3217505	0.004283527843	1.0660032	8.71438533e-06
0.3799554	8.71438533e-06	1.1284221	6.95707276e-05
0.4423742	6.95707276e-05	1.1908409	8.71438533e-06
0.5047930	8.71438533e-06	1.2490457	0.004283527843
0.5667292	5.52195124e-04	1.3072506	8.71438533e-06
0.6286654	8.71438533e-06	1.3696694	6.95707276e-05
0.6910842	6.95707276e-05	1.4320882	8.71438533e-06
0.7535030	8.71438533e-06	1.4940244	5.52195124e-04

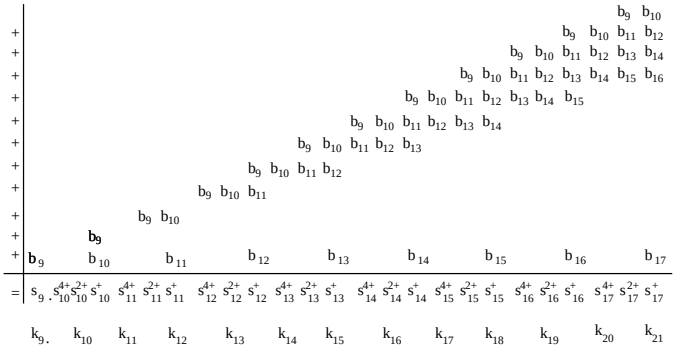

Figure 5. The computation of the coefficients s_j to compute the remaining error ϵ_r.

binary word k by using Booth-encoding. Hence, the partial sums are selected by the Booth-encoded digits k_l, where $k_l \in \bar{2}, \bar{1}, 0, 1, 2$ and the remaining error ϵ_r is computed according to

$$\epsilon_r = \sum_{l=9}^{21} k_l \cdot \epsilon_9. \tag{15}$$

Since k is required to be in a minimally redundant radix-4 representation, the two rows of full-adders and the CPA can be replaced by a minimally redundant radix-4 adder leading to a critical path of six full-adders. The multiplication of $k_l \cdot \epsilon_9$ requires additional 12 rows of full-adders. The final result of ϵ_r is added to d using a redundant adder to d. Note that the largest value of ϵ_r is less than $1.5 \cdot \epsilon_9 < 2^{-31}$. Hence, the addition of ϵ_r only influences the directions of the micro-rotation for iterations with $i > 30$.

Table 3. The computation of the remaining discontinuity ϵ_r

$bp_{diff,11-17}$	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}
0000000	0	0	0	0	0	0	0
0000001	0	0	0	0	0	0	1
0000010	0	0	0	0	0	1	1
0000011	0	0	0	0	0	1	2
0000100	0	0	0	0	1	1	2
0000101	0	0	0	0	1	1	3
0000110	0	0	0	0	1	2	3
0000111	0	0	0	0	1	2	4
0001000	0	0	0	1	1	2	4
0001001	0	0	0	1	1	2	5
0001010	0	0	0	1	1	3	5
0001011	0	0	0	1	1	3	6
0001100	0	0	0	1	2	3	6
0001101	0	0	0	1	2	3	7
0001110	0	0	0	1	2	4	7
0001111	0	0	0	1	2	4	8
0010000	0	0	1	1	2	4	8
0010001	0	0	1	1	2	4	9
0010010	0	0	1	1	2	5	9
0010011	0	0	1	1	2	5	10
0010100	0	0	1	1	3	5	10
0010101	0	0	1	1	3	5	11
0010110	0	0	1	1	3	6	11
0010111	0	0	1	1	3	6	12
0011000	0	0	1	2	3	6	12
0011001	0	0	1	2	3	6	13
0011010	0	0	1	2	3	7	13
0011011	0	0	1	2	3	7	14
0011100	0	0	1	2	4	7	14
0011101	0	0	1	2	4	7	15
0011110	0	0	1	2	4	8	15
0011111	0	0	1	2	4	8	16

2.2 The rotation mode in hyperbolic coordinate systems

In [10, 14], no correlation between the rotation angle θ and the directions of the micro-rotations for hyperbolic coordinate systems is reported. Similar to the circular coordinate systems, there is also 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 is chosen to be the same as for the circular coordinate systems (5).

Due to the incomplete representation of the hyperbolic rotation angle θ_i, some iterations have to be performed twice. In [2], it was recommended that every 4^{th}, 13^{th}, $(3k+1)^{th}$ iteration should be repeated to complete the angle representation. Table 4 shows the incomplete relationship of the angles θ_i. Just as in the case for circular coordinate systems, a correlation between representation of d and the input angle θ can be obtained (see Fig. 7). The figure shows the correlation from angles in the range of -1.05 to 1.05. By zooming into Fig. 7 (see Fig. 8) a piece-wise linear relationship including discontinuities can be recognized. The slope of the difference between the sought word d and the angle θ is either -1 or -0.5 depending on θ. The slope of -1 is caused by the fact that in the hyperbolic coordinates the first iteration starts with $k = 1$ compared to $k = 0$ for circular coordinates. This leads to the relationship shown in (16). Hence, it does not require a multiplication but only one shift-add operation. By using these results following correlation can be obtained for the computation of the directions of the micro-rotations

$$d = \theta - t \cdot 0.5 \cdot \theta + b + \epsilon, \tag{16}$$

where b represents the positive and negative value of d (depending on a positive or negative θ) at the point $\theta = 0$, ϵ is

Table 4. The rotation angles in hyperbolic coordinates

k	θ_k	$\sum_{i=k+1}^{\infty} \theta_i$	$\Delta_k = \theta_k - \sum_{i=k+1}^{\infty} \theta_i$
1	0.54930614433	0.50616322940	0.04314291493
2	0.25541281188	0.25075041751	0.00466239436
3	0.12565721414	0.12509320337	0.00056401076
4	0.06258157147	0.06251163190	0.00006993957
5	0.03126017849	0.03125145341	0.00000872508
6	0.01562627175	0.01562518165	0.00000109009
7	0.00781265895	0.00781252270	0.00000013624
8	0.00390626986	0.00390625283	0.00000001703
9	0.00195312748	0.00195312535	0.00000000212
10	0.00097656281	0.00097656255	0.00000000026
11	0.00048828128	0.00048828125	0.00000000003

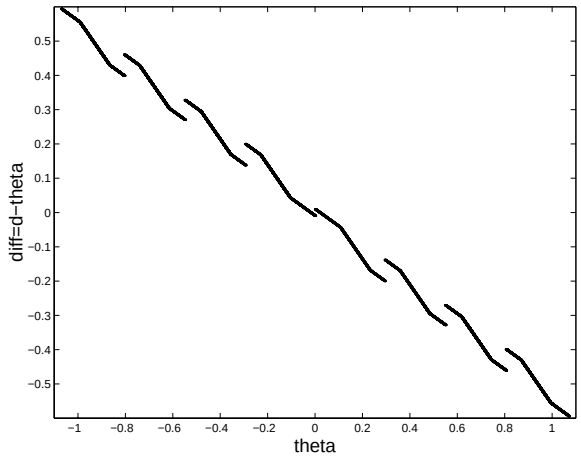

Figure 6. The correlation for the rotation mode between the rotation directions and the given rotation angle θ for hyperbolic coordinate systems. The figure shows the range from -1.05 to 1.05.

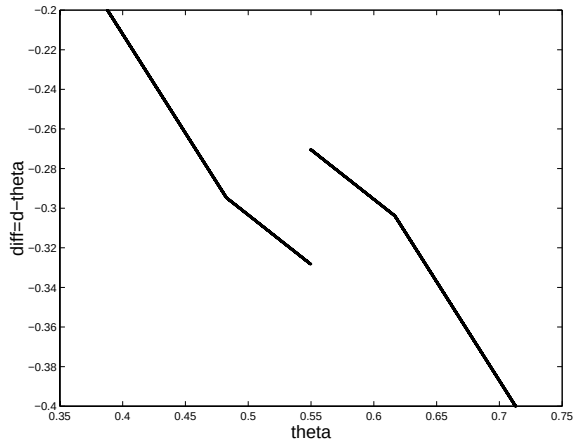

Figure 7. The correlation for the rotation mode between the rotation directions and the given rotation angle θ for hyperbolic coordinates. The figures show the range from 0.35 to 0.75.

the sum of all discontinuities in the graph from 0 to θ and t corresponds to either 0 or 1 depending on value of θ. Like in the circular coordinates, the computation of the first two terms in (16) is trivial while the computation of ϵ is slightly more involved.

2.2.1 Characteristics of discontinuities

The break points of the graph shown in Fig. 7 can be analytically calculated. The discontinuities are caused by the same reason as already described for the circular coordinate systems and are located at

$$bp_{rm,i} = g_1\mathrm{atanh}(2^{-1}) + \mathrm{g_2atanh}(2^{-2}) + ... + \mathrm{g_i atanh}(2^{-\mathrm{i}}), \quad (17)$$

where g_i are either $1, 0$ or -1. However, in the hyperbolic coordinates, the discontinuities only appear in the regions with the slope of -0.5. To compensate for the discontinuities, a similar strategy as in the rotation mode in circular coordinate systems can be explored. Nevertheless, in hyperbolic coordinate systems, three regions have to be distinguished, regions with the slope $m = -1$ and $m = -0.5$. The latter regions has to be split into two regions due to the discontinuities. Hence, two comparisons have to be performed. The architecture of the CORDIC algorithm in hyperbolic coordinate sytems has to include one more fast adder.

High Precision It is not possible to reduce the size of the ROM like in the rotation mode for circular coordinate systems. Since there is a region of discontinuities and not a discrete jump, the maximal error introduced by the remaining discontinuity ϵ_r corresponds to half the width of the discontinuity region.

3 The Novel CORDIC Architecture

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 LSB one can be added while performing $\theta \pm bp_i$. Note that the missing LSB one does not influence the correct selection of ϵ. 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/adds the breakpoint bp_i from 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 argu-

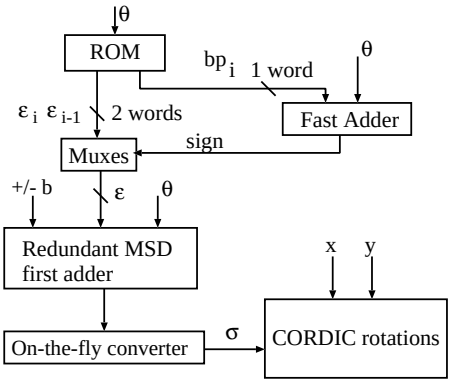

Figure 8. The novel architecture for the proposed CORDIC algorithm (rotation mode)

ments $\arctan(2^{-i})$, where

$$\arctan(2^{-i}) < \sum_{k=i+1}^{i+l} \arctan(2^{-k}), \qquad (18)$$

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 $l-1$ 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 5 shows the maximum number of consecutive unidirectional rotations depending on the iteration number 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 (see Fig. 3.1).

Table 5. The maximal number of consecutive rotation in the same direction

i	$l-1$	i	$l-1$	i	$l-1$
0	3	2	6	4	10
1	5	3	8	5	12

4 Conclusion

This paper has presented for the first time a CORDIC algorithm to compute the directions of the required micro-rotation 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 or 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 cycle corresponds to the delay of twelve full-adders. This corresponds to a speed-up of about 20% and 30% to previous reported CORDIC implementations in rotation and vectoring mode. The implementation is suitable for word-lengths up to 54 bits and 19 bits for rotation and vectoring mode, respectively, while maintaining 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, "Constant Scale Factor, On-Line CORDIC Algorithm in the Circuar Coodinate System," in *VLSI Signal Processing*, (Sakai, Japan), pp. 562–571, Oct. 1995.

[8] J. Duprat and J.-M. Muller, "The CORDIC Algorithm: New results for Fast VLSI Implementation," *IEEE Transcation 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 Transaction on Computers*, vol. 45, pp. 307–318, Mar. 1996.

[10] S. Wang, V. Piuri, and E. Swartzlander, "Hybrid CORDIC Algorithms," *IEEE Transcation 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] M. Kuhlmann and K. K. Parhi, "A High-Speed CORDIC Algorithm and Architecture for DSP Applications," in *Proc. of the 1999 IEEE Workshop on Signal Processing Systems*, (Taipei, Taiwan), Oct. 1999.

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

[17] K. K. Parhi, "Fast low-energy VLSI binary addition," in *Proc. of IEEE Int. Conf. on Computer Design, (ICCD)*, (Austin, TX), pp. 676–684, Oct. 1997.