EURASIP Journal on Applied Signal Processing 2002:9, 936–943

P-CORDIC: A Precomputation Based Rotation CORDIC Algorithm

Martin Kuhlmann
Broadcom Corporation, Irvine, CA 92619, USA
Email: kuhlmann@broadcom.com

Keshab K. Parhi
Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA
Email: parhi@ece.umn.edu

Received 30 August 2001 and in revised form 14 May 2002

This paper presents a CORDIC (coordinate rotation digital computer) algorithm and architecture for the rotation mode 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. The algorithm is capable to perform the CORDIC computation for an operand word-length of 54 bits. 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.

Keywords and phrases: CORDIC, computer arithmetic, constant scale factor, precomputation, rotation mode.

1. INTRODUCTION

CORDIC (coordinate rotation digital computer) [1, 2] is an iterative algorithm for the calculation of the rotation of a 2-dimensional vector, in linear, circular, or hyperbolic coordinate systems, using only add and shift operations. It has a wide range of applications including discrete transformations such as Hartley transform [3], discrete cosine transform [4], fast Fourier transform (FFT) [5], chirp Z transform (CZT) [6], solving eigenvalue and singular value problems [7], digital filters [8], Toeplitz system and linear system solvers [9], and Kalman filters [10]. It is also able to detect multiuser in code division multiple access (CDMA) wireless systems [11].

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^*) (see Figure 1). In every micro-rotation i, fixed angles of the value $\arctan(2^{-i})$ 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. 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

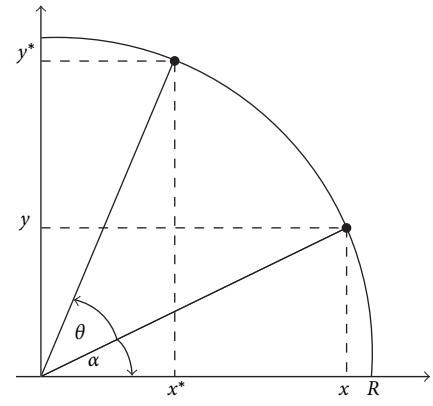

Figure 1: The rotation and vectoring mode of the CORDIC algorithm.

the CORDIC rotations are as follows:

$$\begin{aligned} x_{i+1} &= x_i + m \cdot \sigma_i \cdot 2^{-i} \cdot y_i, \\ y_{i+1} &= y_i - \sigma_i \cdot 2^{-i} \cdot x_i, \\ z_{i+1} &= z_i - \frac{1}{m} \cdot \sigma_i \cdot \arctan\left(\sqrt{m} 2^{-i}\right), \end{aligned} \tag{1}$$

where σ_i is the weight of each micro-rotation 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, the scale factor is constant and can be precomputed according to

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

The computation of the scale factor can be truncated after $n/2$ 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_1,

$$K_1 = \prod_{i=0}^{n/2} \left(1 + \sigma_i^2 \cdot 2^{-2i}\right)^{1/2}. \tag{3}$$

There are two different approaches for the computation of the CORDIC algorithm. The first one uses consecutive rotations in positive and/or negative direction, where the weight of each rotation is 1. Hence, σ_i is either -1 or 1, depending on the sign of the angle remainder $z(i)$. In every iteration a significant amount of time is used to examine the most significant bit in case of a binary architecture or the most significant three digits of a redundant architecture to predict the sign of $z(i)$ and hence the rotation direction σ_i. In comparison to the CORDIC implementations with constant scale factor, other implementations use a minimally redundant radix-4 or an even higher radix number representation [12, 13, 14]. These architectures make use of a wider range of σ_i. In case of a minimally redundant radix-4 architecture, $\sigma_i \in \{-2, -1, 0, 1, 2\}$. By using this numbering system, the number of iterations can be reduced. However, the computation time per iteration increases, since it takes more time to differentiate between five different rotation direction values and to generate five different multiples of $\arctan(2^{-i})$. The scale factor also becomes variable and has to be computed every time, due to the absence of consecutive rotations leading to an increase in area.

To speed up the computation time of the CORDIC algorithm, either the number of iterations or the delay of each iteration have to be minimized. The proposed algorithm introduces a novel approach, in which the rotation direction can be precomputed by adding the rotation angle θ, a constant and a variable adjustment which is stored in a table. Hence, a significant speedup of the delay per iteration is obtained. Since all rotation directions are known before the actual rotation begins, more than one rotation can also be performed in one iteration leading to a reduction in latency. The proposed architecture also eliminates the z-datapath and reduces the area of the implementation.

This paper is organized as follows. Section 2 presents the theoretical background for the novel CORDIC algorithm for rotation mode and Section 3 presents the novel architecture. Section 4 performs an evaluation of different CORDIC architectures while Section 5 concludes the paper.

2. THE NOVEL CORDIC ALGORITHM

2.1. Mathematical derivation using Taylor series

The summation of all micro-rotation with their corresponding weight σ_i is equivalent to the rotation angle θ

$$\theta = \sum_{i=0}^{n} \sigma_i \cdot \arctan\left(2^{-i}\right), \tag{4}$$

where $\sigma_i \in \{-1, 1\}$, corresponding to the addition and subtraction of the micro-angles θ_i. Since consecutive rotations are employed, the scale factor is constant. The value of σ can be interpreted as a number in radix-2 representation. The goal of the proposed method is to compute the sequence of the micro-rotation without performing any iteration. To accomplish this, σ_i is recoding as $2d_i - 1$ leading to a binary representation in which a zero corresponds to the addition of a micro-angle [15, 16]. This allows the use of simple binary adders. Adding and subtracting 2^{-i} to (4) results in

$$\theta = \sum_{i=0}^{\infty} \left(2d_i - 1\right) \cdot \left(2^{-i} - 2^{-i} + \arctan\left(2^{-i}\right)\right) \tag{5}$$

$$= \sum_{i=0}^{\infty} \left(2d_i - 1\right) \cdot 2^{-i} - \sum_{i=0}^{\infty} \left(2d_i - 1\right) \cdot \left(2^{-i} - \arctan\left(2^{-i}\right)\right) \tag{6}$$

$$= 2d - 2 + \sum_{i=0}^{\infty} \left(2^{-i} - \arctan\left(2^{-i}\right)\right) - \sum_{i=0}^{\infty} 2d_i \left(2^{-i} - \arctan\left(2^{-i}\right)\right) \tag{7}$$

$$= 2d - c_1 - \operatorname{sign}(\theta) \cdot 2\left(1 - \arctan(1)\right) - \sum_{i=1}^{\infty} 2d_i \left(2^{-i} - \arctan\left(2^{-i}\right)\right), \tag{8}$$

where c_1 corresponds to $c_1 = 2 - \sum_{i=0}^{\infty} (2^{-i} - \arctan(2^{-i}))$.

Solving (8) for d results in

$$\begin{aligned} d &= 0.5\theta + 0.5c_1 + \operatorname{sign}(\theta) \cdot \left(1 - \arctan(1)\right) - \sum_{i=1}^{\infty} d_i \cdot \left(2^{-i} - \arctan\left(2^{-i}\right)\right) \\ &= 0.5\theta + c + \operatorname{sign}(\theta) \cdot \epsilon_0 - \sum_{i=1}^{\infty} d_i \cdot \epsilon_i, \end{aligned} \tag{9}$$

where c corresponds to $0.5c_1$. Table 1 shows the values of the partial offsets ϵ_i for the first 10 values of i and indicates that the value of ϵ_i decreases approximately by a factor of 8 with increasing i. Hence, the summation of $d_i\epsilon_i$ can be limited to

$$\begin{aligned} d &= 0.5\theta + c - \operatorname{sign}(\theta) \cdot \epsilon_0 - \sum_{i=1}^{n/3} d_i \cdot \epsilon_i, \\ d &= 0.5\theta + c - \operatorname{sign}(\theta) \cdot \epsilon_0 - \delta. \end{aligned} \tag{10}$$

Table 1: The values of ϵ_i of the first 10 values of i.

Iteration i	Partial offset ϵ_i
0	0.214601836602551690
1	3.635239099919176e-02
2	5.021336873135844e-03
3	6.450054532385649e-04
4	8.119000404265152e-05
5	1.016656973172375e-05
6	1.271379523169197e-06
7	1.589398988887035e-07
8	1.986803302817237e-08
9	2.483521181314878e-09

Rather than storing the partial offsets ϵ_i and computing the sum over all i of the product $d_i\epsilon_i$, $\delta = \sum_{i=1}^{n/3} d_i\epsilon_i$ can be precomputed and stored. Hence, the only difficulty consists of determining which offset corresponds to the input θ. This can be achieved by comparing the input θ with a reference angle θ_{ref}. The reference angles θ_{ref} correspond to the summation of the first $n/3$ micro-rotation. To be certain to obtain the correct offset, θ has to be larger than the reference angle θ_{ref}. All reference angles are stored in a ROM and are accessed by the most significant $n/3$ bits of θ. In addition to the reference angles, the values of δ are stored. In case of a negative difference $\theta_{\text{ref}} - \theta$, the corresponding δ is selected, otherwise the next smaller value of δ is chosen to be subtracted from $\theta + c - \text{sign}(\theta) \cdot \epsilon_0$.

Example 1. Assuming we have a word-length of 16 bits and $\theta = 0.9773844$. According to Table 2, θ_{ref} corresponds to 0.97337076 and $\delta = 0.03644375$. Hence, d is computed as

$$\begin{aligned} d &= 0.5 \cdot \theta + 1 - 0.5 \cdot c + \sum_{i=0}^{n/3} d_i\epsilon_i \\ &= 0.5 \cdot 0.9773844 + 1.08624513 + 0.03644375 \\ &= 1.6113811 = 1.1001110010000011_2, \\ \sigma &= 11\bar{1}\bar{1}111\bar{1}\bar{1}1\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}11. \end{aligned} \tag{11}$$

2.2. High precision

By using a mantissa of $n = 54$ bits (corresponding to the floating point precision), the ROM for storing all offsets would require 2^{18} entries. This is rather impractical since the required area to implement the ROM will exceed by far the area for the CORDIC implementation. To reduce the area for the ROM, δ can be split into two parts,

$$\delta = \delta_{\text{ROM}} + \delta_r, \tag{12}$$

where δ_{ROM} is stored in a ROM while δ_r is computed. By examining the Taylor series expansion of $\arctan(2^{-i})$, it becomes obvious that the partial offset ϵ for iteration i and $i+1$ corresponds to

$$\epsilon_i = 2^{-i} - \arctan\left(2^{-i}\right) = \left(-\frac{2^{-3i}}{3} + \frac{2^{-5i}}{5} - \frac{2^{-7i}}{7} + \frac{2^{-9i}}{9} - \cdots \right), \tag{13}$$

$$\epsilon_{i+1} = 2^{-i-1} - \arctan\left(2^{-i-1}\right) \tag{14}$$

$$= \left(-\frac{2^{-3i-3}}{3} + \frac{2^{-5i-5}}{5} - \frac{2^{-7i-7}}{7} + \frac{2^{-9i-9}}{9} - \cdots \right) \tag{15}$$

$$= 2^{-3}\left(-\frac{2^{-3i}}{3} + \frac{2^{-5i-2}}{5} - \frac{2^{-7i-4}}{7} + \frac{2^{-9i-6}}{9} - \cdots \right). \tag{16}$$

By comparing (13) and (16), it can be seen that (13) is about 2^3 times larger than (16). Assuming a word-length of n bits and $i > \lceil n/5 \rceil - 2$, the factor is 2^3. Hence, the term $\epsilon_{\lceil n/5 \rceil - 1} = -2^{-3(\lceil n/5 \rceil - 1)}/3 + 2^{-5(\lceil n/5 \rceil - 1)}/5$ can be stored in a ROM and

Table 2: The reference angles of the rotation mode and their corresponding values of δ for an operand word-length of 16 bits.

θ_{ref}	δ
−0.01640412	0.00008119
0.04607554	0.00009136
0.10746824	0.00064501
0.16994791	0.00065517
0.23230586	0.00072620
0.29478553	0.00073636
0.34871558	0.00502134
0.41119525	0.00503150
0.47355320	0.00510253
0.53603287	0.00511269
0.59742557	0.00566634
0.65990524	0.00567651
0.72226319	0.00574753
0.78474286	0.00575770
0.78605347	0.03635239
0.84853314	0.03636256
0.91089109	0.03643358
0.97337076	0.03644375
1.03476346	0.03699740
1.09724313	0.03700756
1.15960108	0.03707859
1.22208075	0.03708875
1.27601080	0.04137373
1.33849046	0.04138389
1.40084842	0.04145492
1.46332808	0.04146508
1.52472079	0.04201873
1.58720045	0.04202890

the remaining offset δ_r is computed as

$$\delta_r = \sum_{j=\lceil n/5\rceil-1}^{\lceil n/3\rceil} d_j \cdot \epsilon_{\lceil n/5\rceil-1} \cdot 2^{-3(j-\lceil n/5\rceil+1)} < 2^{-3(\lceil n/5\rceil-1)}. \quad (17)$$

The largest magnitude of δ_r is smaller than $2^{-3(\lceil n/5\rceil-1)}$.

Example for high precision

Assume that we have a word-length of 50 bits and $\theta = 0.977384381116$. Using the most significant 9 bits of θ, $\delta_{\text{ROM}} = 0.03644501895249$ can be obtained. Hence, d is computed according to

$$\begin{aligned} d &= 0.5 \cdot \theta + 1 - 0.5 \cdot c + \delta_{\text{ROM}} + \delta_r \\ &= 0.5 \cdot 0.97738438111600 + 1.08624514683872 \\ &\quad + 0.03644501895249 + 2.483521181241566e \\ &\quad - 09 \cdot \left(1 + 2^{-18} + 2^{-21} + 2^{-24}\right) \\ &= 1.61138235883274 \\ &= 1.10011100100000111000011011 \\ &\qquad 11001001000100110111000012. \end{aligned} \quad (18)$$

2.3. The rotation mode in hyperbolic coordinate systems

Similar to the circular coordinate system, a simple correlation between the input angle θ and the directions of the micro-rotation can be obtained. 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 4th, 13th, $(3k+1)$th iteration should be repeated to complete the angle representation.

Similar to the rotation mode in circular coordinate system, the rotation angle θ is equivalent to the summation of all micro-rotation with their corresponding weight. This leads to

$$\begin{aligned} \theta = \sum_{i=0}^{\infty} \sigma_i \cdot \operatorname{arctanh}\left(2^{-i}\right) + \sigma_4^{\text{extra}} \cdot \operatorname{arctanh}\left(2^{-4}\right) \\ + \sigma_{13}^{\text{extra}} \cdot \operatorname{arctanh}\left(2^{-13}\right) + \sigma_{40}^{\text{extra}} \cdot \operatorname{arctanh}\left(2^{-40}\right) + \cdots. \end{aligned} \quad (19)$$

Performing a Taylor series expansion and applying $\sigma_i = 2d_i - 1$ results in

$$\begin{aligned} d &= 0.5\theta + 0.5c_1 + \operatorname{sign}(\theta) \cdot 2\left(0.5 - \operatorname{arctanh}(0.5)\right) \\ &\quad + \left(d_4^{\text{extra}} - \frac{1}{2}\right) \cdot \operatorname{arctanh}\left(2^{-4}\right) \\ &\quad + \left(d_{13}^{\text{extra}} - \frac{1}{2}\right) \cdot \operatorname{arctanh}\left(2^{-13}\right) \\ &\quad + \left(d_{40}^{\text{extra}} - \frac{1}{2}\right) \cdot \operatorname{arctanh}\left(2^{-40}\right) \\ &= 0.5\theta + c + d_4^{\text{extra}} \cdot \operatorname{arctanh}\left(2^{-4}\right) + d_{13}^{\text{extra}} \cdot \operatorname{arctanh}\left(2^{-13}\right) \\ &\quad + d_{40}^{\text{extra}} \cdot \operatorname{arctanh}\left(2^{-40}\right), \end{aligned} \quad (20)$$

where c corresponds to $c = 1 - 0.5\sum_{i=1}^{\infty}(2^{-i} - \operatorname{arctanh}(2^{-i})) - 0.5 \cdot (\operatorname{arctanh}(2^{-4}) + \operatorname{arctanh}(2^{-13}) + \operatorname{arctanh}(2^{-40}) + \cdots)$. Since these extra rotation are not known in advance, an efficient high precision VLSI implementation is not possible. However, for signal processing applications using a word-length of less than 13 bits, the ROM size corresponds to only 14 entries.

3. THE NOVEL ROTATION-CORDIC ARCHITECTURE

For an implementation with the operand word-length of n bits, the pre-processing part consists of a ROM of $2^{\lceil n/5-2\rceil}$ entries in which the reference angles θ_{ref} and the corresponding offsets δ are stored, respectively (see Figure 2). To avoid a second access to the ROM in case of $\theta_{\text{ref}} > \theta$ the next smaller offset δ_{k-1} is additionally stored in the kth entry of the ROM. The ROM is accessed by the $\lceil n/5-2\rceil$ MSB bits of θ. A binary tree adder computes, whether θ is smaller or larger than the chosen reference angle θ_{ref} and selects the corresponding offset (either δ_k or δ_{k-1}). Using a 3 : 2 compressor and another fast binary tree adder, the two required additions to obtain $d_{\text{approx}} = 0.5\theta + c_2 + \delta_{\text{ROM}}$ can be performed, where c_2 corresponds to $c + \operatorname{sign}(\theta)\epsilon_0$. Using the bits $d_{\lceil n/5\rceil-1}$ to $d_{\lceil n/3\rceil}$, δ_r can be computed according to (17) and has to be added to d_{approx}. For the worst case scenario, there is a possible ripple from the bit $d_{3(\lceil n/5\rceil-1)}$ to the bit $d_{(\lceil n/5\rceil)}$ which would call for a time consuming ripple adder. However, by employing an extra rotation for $d_{3(\lceil n/5\rceil-1)-1}$ this limitation can be resolved. This extra rotation corresponds to the overflow bit of the addition from the bits $d^{\text{approx}}_{-3(\lceil n/5\rceil-1)\cdots n}$ and δ_r. The additional rotation also does not affect the scale factor, since $3(\lceil n/5\rceil - 1) > n/2$. For a precision of $n \leq 16$ bits, there are less than 32 offsets which can be stored in a ROM and the additional overhead to compute δ_r can be removed.

The alternative architecture can be chosen by realizing that the directions of the micro-rotations are required in a most significant bit first manner (see Figure 2). As in the previous architecture, a fast binary adder is employed to determine which offset has to be selected. A redundant sign digit adder adds 0.5θ, c, and δ_{ROM} and an on-the-fly converter starts converting resulting into the corresponding binary representation. Normally, the most significant bit cannot be determined until the least significant digit is converted. However, such worst cases do not exist in the CORDIC implementation, due to the redundant representations of the angles $\arctan(2^{-i})$, where

$$\arctan\left(2^{-i}\right) < \sum_{k=i+1}^{\infty} \arctan\left(2^{-k}\right), \quad (21)$$

as opposed to the binary representation

$$2^{-i} > \sum_{k=i+1}^{\infty} 2^{-k}. \quad (22)$$

Therefore, 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 lth

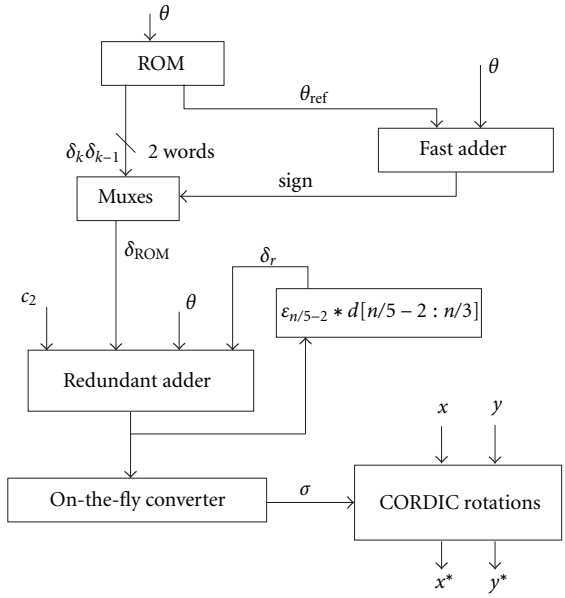

Figure 2: The novel architecture for the rotation mode.

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

i	θ_i	$l-1$	i	θ_i	$l-1$	i	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

iteration has to be rotated into the opposite direction. This happens if the angle remainder $z_i \approx 0$. Table 3 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 online converters and its most significant bits can already be used to start the rotations in the x/y datapath.

Example 2. Assuming an angle $\theta = 0.001$. Hence, the angle remainder θ_i correspond to

$$\begin{aligned}
\theta_0 &= 0.001,\\
\theta_1 &= \theta_0 - \arctan(1) = -0.7844,\\
\theta_2 &= \theta_1 + \arctan(0.5) = -0.3208,\\
\theta_3 &= \theta_2 + \arctan(0.25) = -0.0758,\\
\theta_4 &= \theta_3 + \arctan(0.125) = 0.0486.
\end{aligned} \tag{23}$$

The next rotation has to be performed in the negative directions, since $\theta_4 > 0$. Hence, it is not possible to obtain rotation sequence like $\sigma_{0\cdots4} = 01111$ but it has to be $\sigma_{0\cdots4} = 01110$.

3.1. Evaluation of the z-datapath

Delay analysis

In this paper, we assume a similar delay model as proposed in [14]. Nevertheless in [14], the unit delay is set to a gate delay while in our evaluation the unit delay is set to a full-adder delay. Hence, the delays for 2-input (NAND, NOR) gate, XOR, multiplexer, register, and full-adder are 0.25, 0.5, 0.5, 0.5, and $1t_{\mathrm{FA}}$.

The determination of which offset has to be chosen consists of the delay of the decoder, the ROM, a fast binary n-bit tree adder and a multiplexer. Assuming a delay of $\log_2(m)$ gate delays for the decoder, where m corresponds to the number of rows in the ROM ($m < \log_2(n) + 1$), one for the word-line driver and another for the ROM, $\log_2(n) \cdot t_{\mathrm{Mux}}$ for the fast binary adder and $0.5 \cdot t_{\mathrm{FA}}$ for the multiplexer, we can obtain the correct value of δ_{ROM} after a delay of $(0.5\log_2(n) + 1 + 0.25\log_2(\log_2(n))) \cdot t_{\mathrm{FA}}$.

A 3 : 2 compressor can be employed to reduce the number of partial products to two. An additional fast binary tree adder can compute the final value of d_{approx}. Hence, the entire delay to obtain d_{approx} corresponds to

$$\begin{aligned}
&\Big(0.5\log_2(n) + 1 + 0.25\log_2\left(\log_2(n)\right)\\
&\qquad + 1 + 0.5\log_2(n)\Big) \cdot t_{\mathrm{FA}} = \left(\log_2(n) + 2.25\right) \cdot t_{\mathrm{FA}}.
\end{aligned} \tag{24}$$

After obtaining the bits $d_{\lceil n/5\rceil-1}$ to $d_{\lceil n/3\rceil}$, δ_r can be computed. Since the value of δ_r is smaller than $2^{-3(\lceil n/5\rceil-1)}$ and the value of $d_{\mathrm{approx}} + \delta_r$ is not required before $2^{3(\lceil n/5\rceil)}t_{\mathrm{FA}}$ the computation of δ_r is not in the critical path.

Alternatively to the 3 : 2 compressor and the tree adder, a minimally redundant radix-4 sign digit adder can be employed which has a delay of two full-adders. Hence, all output digits are available after these two full-adder delays. An additional on-the-fly converter converts the digits into its equivalent binary representation starting with the MSD. It requires a delay of multiplexer and four NANDs/NORs to convert one digit which results in $1.5t_{\mathrm{FA}}$ per digit (1 digit = 2 bits). The last digit is converted after a delay of $(n/2 + 1) \cdot 1.5t_{\mathrm{FA}}$. As already described in Table 3, bit $n/3$ is stable as soon as the last digit (corresponding to bit n) has been converted. Hence, the $n/3$ rotation can be performed after a delay of $(n/2 + 1) \cdot 1.5t_{\mathrm{FA}}$. Therefore, the iterations $i = 0$ can already be performed after a delay of $(n/2 + 1) \cdot 1.5t_{\mathrm{FA}} - n/3 \cdot 2t_{\mathrm{FA}} = (1/12 \cdot n + 1)t_{\mathrm{FA}}$. Note that the conversion of one redundant digit is performed faster than the addition/subtraction of the x/y datapath. Hence, an initial delay of $(1/12 \cdot n + 1)t_{\mathrm{FA}} + (\log_2(n) + 2.25)t_{\mathrm{FA}} = (1/12 \cdot n + \log_2(n) + 3.25)t_{\mathrm{FA}}$ has to be added to the delay of the x/y datapath.

Area analysis

Previously, the area of the z-datapath consists of $n/2$ iterations in which $(n+\log_2 n+2)$ multiplexers and $(n+\log_2 n+2)$ full-adders and registers are employed. Additionally, due to the Booth encoding, in the last $n/4$ iterations, about $2(n + \log_2 n + 2)$ multiplexers and $(n + \log_2 n + 2)$ full-adders are required. Assuming $A_{\mathrm{FA}} = 1.93 \cdot A_{\mathrm{mux}}$ and $A_{\mathrm{FA}} = 1.61 \cdot A_{\mathrm{reg}}$ (values are based on layouts), the hardware complexity of the z-datapath results in $A_z = 1.7 \cdot n(n+\log_2 n+2)A_{\mathrm{FA}}$. Assuming a word-length of 54 bits and neglecting the required area for

the examination of the most significant three digits, about $5700A_{\mathrm{FA}}$ are required.

The proposed architecture utilizes a ROM of word-length n and $2^{\lceil n/5-2\rceil}$ entries, requiring an area of $n \cdot 2^{\lceil n/5-2\rceil} \cdot A_{\mathrm{FA}} \cdot 1/50$ resulting in $552A_{\mathrm{FA}}$ for a word-length of 54 bits. The implementation of the decoders can be done in multiple ways. NOR based decoders with precharge lead to the fastest implementation. However, the decoder area becomes larger. The decoder size per word-line corresponds to $A_{\mathrm{dec}} = 0.83A_{\mathrm{FA}}$. Since $2^{\lceil n/5\rceil-2}$ decoders are required, the area for all decoder corresponds to $A_{\mathrm{dec,total}} = 0.83 \cdot 2^{\lceil\lceil n/5\rceil-2\rceil} = 424A_{\mathrm{FA}}$, assuming a 54 bit word-length. The ROM has to store θ_{ref}, δ_k, and δ_{k-1}. This results in a total area for the ROM and the decoder of about $2080A_{\mathrm{FA}}$. The computation of δ_r requires $n/3 - n/5 + 2 = 2n/15 + 2$ rows of CSA (carry-save-adders) and Muxes and a final fast binary tree adder. Note that each row of CSA adders and Muxes only consists of $(n - 3n/5 + 6 = 2n/5 + 6)$ bits (the more significant bits are zero). The required area corresponds to $10 \cdot 27A_{\mathrm{FA}} + 10 \cdot 27A_{\mathrm{mux}}$ and $5 \cdot 27A_{\mathrm{FA}}$, respectively. Hence, the computation of δ_r requires $540A_{\mathrm{FA}}$. Moreover, the two redundant sign digit adder require $2n \cdot A_{\mathrm{FA}}$, while the converter consists of about $(0.5n^2 + n)A_{\mathrm{mux}}$. This corresponds to 108 and $696A_{\mathrm{FA}}$ for a word-length of 54 bits. This makes a total of $3426A_{\mathrm{FA}}$, which is about 60% of the z-datapath previously employed.

3.2. Evaluation of the x/y datapath

In the first $n/2$ micro-rotations, the critical path of the x/y rotator part consists of a multiplexer and a $4:2$ compressor, which has a combined critical path of 2 full-adders. The last $n/2$ micro-rotations can be performed only using $n/4$ iterations, since Booth encoding can be employed. However, the delay of the selection for the multiple of the shifted x/y components requires slightly more time, resulting in a delay of about one full-adder delay. The delay for the $4:2$ compressor remains 1.5 full-adder. Hence, the critical path of the entire x/y rotator part consists of $n/2 \cdot 2t_{\mathrm{FA}} + n/4 \cdot 2.5 \cdot t_{\mathrm{FA}} = 1.625n \cdot t_{\mathrm{FA}}$. Note that the direction of the first iteration is already known; hence, the first iteration is not in the critical path. Therefore, the critical path of the entire x/y rotator part consists of $(1.625n - 2)t_{\mathrm{FA}}$.

As an example, for a word-length of $n = 16$ bits, the x/y datapath delay and the entire delay of the CORDIC algorithm corresponds to 24 and 32.5 full-adder delays, respectively.

3.3. Scale factor compensation

Since the scale factor is constant, the x and y values can already be scaled while the rotation direction is being computed. The scaling requires an adder of word-length $(n + \log_2(n))$ bits. Using a binary tree adder, this results in a delay of $\log_2(n + \log_2(n)) \cdot t_{\mathrm{Mux}}$. For the scale factor, a CSD (canonic signed digit) representation can be used, leading to at most $n/3$ nonzero digits. Applying a Wallace-tree for the partial product reduction, the total delay of the scaling results into $(0.5\log_2(n + \log_2(n)) + \log_{1.5}(n/3)) \cdot t_{\mathrm{FA}} < (1/12 \cdot n + \log_2(n) + 3.25) \cdot t_{\mathrm{FA}} = t_{\mathrm{initial}}$. Hence, the scaling of the x and y coordinates does not affect the total latency of the novel algorithm.

4. OVERVIEW OF PREVIOUSLY REPORTED CORDIC ALGORITHMS

The delay of every iteration can be decomposed into two different time delays, $t_{d,\sigma}$ and $t_{d,xy}$, where $t_{d,\sigma}$ corresponds to the time delay to predict the new rotation direction while $t_{d,xy}$ corresponds to the time delay of the multiplexer/add structure of the x/y datapath. Various implementations have been proposed to obtain a speedup of the CORDIC algorithm. Improvements have been especially made in the reduction of $t_{d,\sigma}$.

In [17], the angle remainder has been decomposed every $k = 3k+1$ iteration. From the given angle θ, the first four rotation directions can be immediately determined. After performing the corresponding addition/subtraction of the terms $\sigma_i \cdot \alpha_i$ from the input angle θ using CSA arithmetic, a fast binary tree adder computes the nonredundant result z_4. The bits 4 to 13 of z_4 deliver the rotation direction σ_4 to σ_{13} which are used to perform the rotation in the x/y datapath and the computation of the next angle remainder z_{40}. Hence, a low latency CORDIC algorithm is obtained. However, a significant reduction in latency is achieved at the cost of an irregular design. Furthermore, it is difficult to perform a $\pi/2$ initial rotation or the rotation of index $i = 0$ for circular coordinates, as it would force a conversion from redundant to conventional arithmetic for the z coordinate just after the first micro-rotation which is costly in time and area. Hence, this parallel and nonpipelined architecture only converges in the range of $[-1, 1]$. The overall latency of this architecture corresponds to about $2n + \log_3(n) + \log_2(n)$ full-adder delay.

In [18], a direct correlation between the z remainder after $\lceil n/3\rceil$ rotations and the remaining rotation direction have been shown. Hence, no more examination of the direction of the micro-rotation has to be performed leading to a considerable reduction in latency. However, in the first $\lceil n/3\rceil$ iteration a conventional method has to be employed.

In [19], the directions of the micro-rotation have been recoded using an offset binary coding (OBC) [20]. The obtained correlation is approximately piecewise linear since small elementary angles can be approximated by $a(i) = \arctan(2^{-i}) \approx s \cdot 2^{n-i-2}$, where s is the slope of the linearity. This is valid for $i \geq m$, where m is an integer which makes the approximation tolerable (normally $m = \lceil n/3\rceil$). Hence, the following correlation can be obtained:

$$\sum_{i=m}^{n-1} \sigma_i 2\alpha_i \approx s \cdot \sum_{i=m}^{n-1} \sigma_i \cdot 2^{-i-1}. \tag{25}$$

By performing some arithmetic computations, the following correlation of the rotation direction can be obtained:

$$\sum_{i=0}^{n-1} \sigma_i \cdot 2^{-i} = \sum_{i=0}^{n-1} \sigma_i \cdot 2^{-i-1} - \sum_{i=m}^{n-1} \sigma_i \cdot \frac{2\alpha(i)}{s}. \tag{26}$$

Hence, a multiplication by the inverse of the slope s is required. This multiplication can be simplified to two stages of addition for an operand word-length of 9 bits. However, in most digital signal processing application, the operands have a word-length of up to 16 bits. Hence, for those applications, the presented method requires more stages of addition to compensate the multiplication resulting in a more complex implementation and an increase in delay.

In [21], a double rotation method is introduced which compensates for the scale factor while performing the regular x/y rotations. However, due to the double rotation nature of this method, $t_{d,xy}$ is increased to about twice its original value.

To reduce the latency of the CORDIC operation, [22] proposed an algorithm using online arithmetic. However, this results in a variable scale factor. This drawback is removed in [23]. In every iteration a significant amount of time is used to examine the most significant three digits to predict σ_i. The employed random logic requires a delay of about 1.5 full-adder delays. Since the x/y datapath consists of a 4-2 compressor, it requires also a delay of 2 full-adders. Hence, the overall iteration delay corresponds to 3.5 full-adder delays. To maintain a constant scale factor, consecutive rotations are required in the first $n/2$, where n corresponds to the word-length of the operands. For the computation of the last $n/2$ bits, Booth encoding can be employed reducing the number of iterations by a factor of 2. However, the selection of multiple of the shifted x and y operands requires an additional multiplexer delay and increases the overall iteration delay to 4 full-adder delays. Hence, the number of iteration is equivalent to $0.75n$ which corresponds to a total latency of $3n$ full-adders (this does not include the scale operation and the conversion).

Other implementations like [24] remove the extra rotations by a branching mechanism in case that the sign of the remainder cannot be determined (most significant three digits are zero). Hence, no extra-rotations are required while the required implementation area is doubled. Nevertheless, the most significant three digits (or most significant six bits) still have to be examined for the prediction of the next rotation direction. In [25], the double step branching CORDIC algorithm is introduced which performs two rotations in a single step. Nevertheless, this method requires an examination of the most significant six digits to detect two rotation directions. Since some of the digits can be examined in parallel, the delay increases only to $2t_{\mathrm{FA}}$. The computation time of a double rotation in the x/y datapath is slightly reduced compared to two normal x/y rotations. Hence, the total amount of computation time corresponds to $0.5n(2t_{\mathrm{FA}} + 3t_{\mathrm{FA}}) = 2.5t_{\mathrm{FA}}$.

In [26], 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 $3n$ full-adders delays.

Table 4: An overview between the proposed algorithm and other CORDIC implementations.

Approach	Delay in t_{FA}
proposed	$1.625n + 1/12 \cdot n + \log_2(n) + 1.25$
[14]	$2n + 6$
[26]	$3n + 1$
[21]	$3.75n$
[27]	$5.25n$
[25]	$2.5n$
[17]	$2n + \log_3(n) + \log_2(n)$

In comparison to the CORDIC implementations with constant scale factor, other implementations use a minimally redundant radix-4 or an even higher radix number representation [12, 13, 14]. By using this number system, the number of iterations can be reduced. However, the prediction of the σ_i becomes more complicated, since there are more possible values for σ_i. In addition, the scale factor becomes variable and has to be computed every time, due to the absence of consecutive rotations. An online computation of the scale factor and a parallel scaling of the x and y operands can be achieved. Depending of the use of CSA or fast carry-propagate-adders (CCLA), the number of iterations can be reduced to $2\lceil n/3 \rceil + 4$ and $n/2 + 1$, respectively. The iteration delay $t_{d,\mathrm{CSA}}$ of the architecture using CSA adders corresponds to the same delay as already described for the last $n/2$ iteration in the constant scale factor using Booth-encoding, while the architecture employing the fast CCLA adders requires $1.5 \cdot_{d,\mathrm{CSA}}$ [14]. Hence, the overall latency of these CORDIC algorithm using a minimally redundant radix-4 digit set corresponds to about $2n$ full-adder delays.

Table 4 provides a delay comparison between the proposed algorithm and other CORDIC implementations. Some of the delays have been taken from [14, 17, 26].

5. CONCLUSION

This paper presented a CORDIC algorithm for the rotation mode which computes the directions of the required micro-rotation before the actual CORDIC computations start while maintaining a constant scale factor. 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. The rotation directions are obtained by adding the rotation angle θ to a constant and a variable offset which is stored in a ROM. An implementation for high precision is also provided which reduces the size of the required ROM. Hence, neither extra or double rotations nor a variable scale factor are required. The implementation is suitable for word-lengths up to 54 bits, while maintaining a reasonable ROM size.

ACKNOWLEDGMENT

This work was supported by the Defense Advanced Research Projects Agency under contract number DA/DABT63-96-C-0050. Prof. Parhi is on leave from the Department of Electrical and Computer Engineering of the University of Minnesota, Minneapolis, MN, USA.

REFERENCES

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. 8, no. 3, pp. 330–334, 1959.

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

[3] L. W. Chang and S. W. Lee, "Systolic arrays for the discrete Hartley transform," *IEEE Trans. Signal Processing*, vol. 39, no. 11, pp. 2411–2418, 1991.

[4] W.-H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Communications*, vol. 25, no. 9, pp. 1004–1009, 1977.

[5] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Trans. on Computers*, vol. 23, no. 10, pp. 993–1001, 1974.

[6] Y. H. Hu and S. Naganathan, "A novel implementation of chirp Z-transform using a CORDIC processor," *IEEE Transaction on Acoustics, Speech, and Signal Processing*, vol. 38, no. 2, pp. 352–354, 1990.

[7] M. Ercegovac and T. Lang, "Redundant and on-line CORDIC: Application to matrix triangularization and SVD," *IEEE Trans. on Computers*, vol. 39, no. 6, pp. 725–740, 1990.

[8] P. P. Vaidyanathan, "A unified approach to orthogonal digital filters and wave digital filters, based on LBR two-pair extraction," *IEEE Trans. Circuits and Systems*, vol. 32, no. 7, pp. 673–686, 1985.

[9] Y. H. Hu and H. M. Chern, "VLSI CORDIC array structure implementation of Toeplitz eigensystem solver," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 1575–1578, April 1990.

[10] T. Y. Sung and Y. H. Hu, "Parallel VLSI implementation of Kalman filter," *IEEE Trans. on Aerospace and Electronics Systems*, vol. 23, pp. 215–224, March 1987.

[11] H. V. Poor and X. Wang, "Code-aided interference suppression for DS/CDMA communications—Part I: Interference suppression capability," *IEEE Trans. Communications*, vol. 45, no. 9, pp. 1101–1111, 1997.

[12] C. Li and S. G. Chen, "A radix-4 redundant CORDIC algorithm with fast on-line variable scale factor compensation," in *International Symposium on Circuits and systems*, pp. 639–642, Hong Kong, June 1997.

[13] R. Osorio, E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapata, "Digit on-line large radix CORDIC rotator," in *Proc. Int. Conf. Application-Specific Array Processors*, pp. 246–257, Strasbourg, France, July 1995.

[14] J. Villalba, J. Hidalgo, E. L. Zapata, E. Antelo, and J. D. Bruguera, "CORDIC architectures with parallel compensation of the scale factor," in *Proc. Int. Conf. Application Specific Array Processors*, pp. 258–269, Strasbourg, France, July 1995.

[15] M. Kuhlmann and K. K. Parhi, "A high-speed CORDIC algorithm and architecture for digital signal processing applications," in *Proc. 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation*, pp. 732–741, Taipei, Taiwan, October 1999.

[16] M. Kuhlmann and K. K. Parhi, "A new CORDIC rotation method for generalized coordinate systems," in *Proc. 1999 Asilomar Conf. on Signals, Systems and Computers*, Pacific Grove, Calif, USA, October 1999.

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

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

[19] S. Nahm and W. Sung, "A fast direction sequence generation method for CORDIC processors," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 635–638, Munich, Germany, April 1997.

[20] N. Demassieux and F. Jutand, *VLSI Implementation for Image Communications*, chapter 7, P. Pirsch, Ed., Elsevier Science, New York, NY, USA, 8th edition, 1993.

[21] 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, no. 9, pp. 989–995, 1991.

[22] H. X. Lin and H. J. Sips, "On-line CORDIC algorithms," *IEEE Trans. on Computers*, vol. 39, no. 8, pp. 1038–1052, 1990.

[23] R. Hamill, J. McCanny, and R. Walke, "On-line CORDIC algorithm and VLSI architecture for implementing QR-array processors," to be published in *Journal of VLSI Signal Processing*, 1999.

[24] J. Duprat and J.-M. Muller, "The CORDIC algorithm: New results for fast VLSI implementation," *IEEE Trans. on Computers*, vol. 42, no. 2, pp. 168–178, 1993.

[25] D. S. Phatak, "Double step branching CORDIC: A new algorithm for fast sine and cosine generation," *IEEE Trans. on Computers*, vol. 47, no. 5, pp. 587–602, 1998.

[26] H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundant implementation without correcting iterations," *IEEE Trans. on Computers*, vol. 45, no. 3, pp. 307–318, 1996.

[27] J.-A. Lee and T. Lang, "A constant-factor redundant CORDIC for angle calculation and rotation," *IEEE Trans. on Computers*, vol. 41, no. 8, pp. 1016–1025, 1992.

Martin Kuhlmann received his Diplome Ingénieur and Ph.D. degrees in electrical engineering from the University of Technology Aachen, Germany in 1997 and from the University of Minnesota in 1999, respectively. Currently, he is a staff design engineer at Broadcom Corporation, Irvine, CA, USA. His research interests include computer arithmetic, digital communication, VLSI design, and deep-submicron crosstalk.

Keshab K. Parhi is a distinguished McKnight University Professor of Electrical and Computer Engineering at the University of Minnesota, Minneapolis, where he also holds the Edgar F. Johnson Professorship. He received the B.Tech., M.S.E.E., and Ph.D. degrees from the Indian Institute of Technology, Kharagpur (India) (1982), the University of Pennsylvania, Philadelphia (1984), and the University of California at Berkeley (1988), respectively. His research interests include all aspects of physical layer VLSI implementations of broadband access systems. He is currently working on VLSI adaptive digital filters, equalizers and beamformers, error control coders and cryptography architectures, low-power digital systems, and computer arithmetic.