Design of a pipelined radix 4 CORDIC processor

J.D. Bruguera a, E. Antelo a and E.L. Zapata b

a Dept. Electrónica. Facultad de Física, Univ. Santiago de Compostela, 15706 Santiago de Compostela, Spain
b Dept. Arquitectura de Computadores, Fac. Informática., Univ. Málaga, 29013 Málaga, Spain

Received 6 April 1992
Revised 16 October 1992

Abstract

In this work we develop a generalization of the CORDIC algorithm for any radix in three coordinate systems, linear, circular and hyperbolic. We carry out a comparative study between different radices at the number of additions level, due to the fact that the complexity in additions determines the total hardware associated with the implementation of the algorithm. Radix 4 minimizes the number of additions, therefore we propose a high speed CORDIC processor based on this radix 4. We have developed new criteria for the vectorization and rotation modes and we introduce a new technique for the compensation of a non-constant scale factor, multifactor compensation. The processor we propose is of a general and pipelined character and uses redundant arithmetic (signed digit) in order to reduce the delay in each stage, parallelizing the operation of the adders using a competitive implementation (reduction of the number of stages) with respect to radix 2 implementations which have recently appeared in the literature.

Keywords. Digital signal processing; CORDIC algorithm; redundant arithmetic; signed-digit adders; pipelining.

1. Introduction

The CORDIC algorithm (COordinate Rotation DIgital Computer), was introduced in 1959 by Volder [19]. In 1971 Walther [20] generalized the algorithm parametrizing it for three different coordinate systems. It is based on the rotation of a vector in a coordinate system which can be linear, circular or hyperbolic. This rotation permits the production of a large number of arithmetic, trigonometric and hyperbolic functions. Its main applications are currently within the field of computational algebra and image processing [10,14]. It is used for matrix inversion, filtering [12], eigenvalue calculations [1], SVD algorithms [5,8,13], orthogonal transforms [2,3,22], graphic applications [21], simulations and function generators [9,17].

The CORDIC is an iterative algorithm which has traditionally been implemented so that all the iterations were executed in the same hardware. Each iteration is carried out only by means of addition/subtraction and shift operations. It is also necessary to have a ROM
memory for storing constants. Due to its iterative nature, the traditional implementation is very slow for high speed computation [5,9]. More recent implementations have introduced the concept of pipelining. The advances in VLSI technology have made this implementations possible, becoming a competitive element for high speed computations [3,11,12,14,21]. The speed of the pipelined CORDIC processor can still be improved by means of the introduction of redundant arithmetic, substituting the conventional adders by carry save [3,8,15] or signed digit adders [13,17,21]. Another type of implementations of the CORDIC algorithm make use of on-line arithmetic [8,13]. These implementations overlay the input and the output of data with the calculation, reducing this way the communications bandwidth of the circuit. In some specific applications (FFT, FCT, etc.), the rotation angles are known a priori, and this permits the design of application specific CORDIC processors. The presentation of the angle in decomposed manner permits the elimination of the hardware associated with iteration z, because it is not necessary to accumulate this angle [2,3,8].

The CORDIC processors proposed in the literature make use of the radix 2 CORDIC algorithm. That is, the rotation angle is decomposed as a sum of powers of two. In this work we generalize the CORDIC algorithm to any radix (power of 2) and present the design of a general application radix 4 pipelined CORDIC rotator. Radix 4 permits the reduction of the number of microrotations. The processor incorporates signed digit redundant arithmetic which drastically reduces the delays in each stage. This delay does not depend on the precision.

The paper is structured in the following way. In Section 2 we describe the radix 2 CORDIC algorithm. In Section 3 generalize the CORDIC algorithm to any radix, presenting general equations. We perform a comparative study between the different radix in order to obtain which is the most adequate in terms of the reduction of hardware complexity. As radix 4 is the optimal, in Section 4 we present the whole algorithm for radix 4 in its rotation and vectorization modes. We propose an architecture for the CORDIC processor in the rotation mode for the three coordinate systems, using redundant arithmetic. We then develop a new technique for the compensation of the scale factor. Finally, in Section 5 we evaluate the architecture we propose and compare it to radix 2 CORDIC processors which have been recently proposed.

2. The radix 2 CORDIC algorithm

The radix 2 CORDIC algorithm [19,20] consists in rotating a vector in a given coordinate system which we parametrize by m (m = 0 linear, m = 1 circular, m = −1 hyperbolic). This rotation is decomposed into a series of prefixed elementary rotations, microrotations, whose sense has to be determined. The number of microrotations is determined by the precision of the algorithm. The microrotation angles are given by \( \alpha_{i,m} = m^{-1/2} \tan^{-1} (m^{1/2} 2^{-i}) \), with \( i = 0,1,\ldots,n-1 \) for circular coordinates and \( i = 1,\ldots,n-1 \) for linear and hyperbolic coordinates, where n is the number of bits of the operands (in the case of hyperbolic coordinates some microrotations have to be repeated in order to insure convergence). An initial rotation of \( \pi/2 \) is necessary for extending the convergence range to \([-\pi,\pi]\) in circular coordinates. The iterative equations that perform each of these microrotations (it is not a perfect rotation) as a function of the parameter m are

\[
\begin{align*}
x_{i+1} &= x_i + m \sigma_i y_i 2^{-i} \\
y_{i+1} &= y_i - \sigma_i x_i 2^{-i} \\
z_{i+1} &= z_i - \sigma_i \alpha_{i,m}
\end{align*}
\]
where \( \sigma_i \in \{+1, -1\} \), is the angle decomposition coefficient, and indicates the sense of each of the microrotations. We have considered that \( m^{-1/2} \tan(m^{1/2} \pi \sigma_{i,m}) = \sigma_i 2^{-i} \). The coordinates obtained with Eq. (1) are scaled by a constant factor \( k_i = (1 + |\sigma_i| m^{-2^{-i}})^{1/2} \) with respect to the coordinates corresponding to an authentic microrotation. The total effect is included in the final scale factor, \( K \), which is given by the product of all the factors introduced in each microrotation:

\[
K = \prod_{i=0}^{n-1} (1 + |\sigma_i| 2^{-2^i})^{1/2}.
\]

The compensation of the scale factor can be achieved with positive and negative scaling [9], that consists in the introduction of the following operation in specific microrotations

\[
x_{i+1,2} = \left(1 + \gamma 2^{-j}\right)x_{i+1,1},
\]

\[
y_{i+1,2} = \left(1 + \gamma 2^{-j}\right)y_{i+1,1}
\]

where \( x_{i+1,1} \) and \( y_{i+1,1} \) are the coordinates of the vector rotated in iteration \( i \) and \( \gamma \in \{-1, +1\} \). The values of \( j \) and \( \gamma \) are selected in order to make the final scale factor one. Another technique is the repetition of microrotations [1], which consists in the introduction of redundant microrotations in order to force the final scale factor to an integer power of two. The last technique of compensation consists in combining the two previous techniques in order to minimize the number of stages [7].

The CORDIC algorithm has two operation modes, rotation and vectorization. In the rotation mode, we force the final value of the \( z \) coordinate to zero, taking \( \sigma_i = \text{sign}(z_i) \). With this mode the addition and multiplication (linear coordinates) sine and cosine (circular coordinates) hyperbolic sine and cosine (hyperbolic coordinates) functions are generated. In the vectorization mode, the final value of coordinate \( y \) is forced to zero, taking \( \sigma_i = \text{sign}(y_i) \). In this mode the CORDIC algorithm generates the subtraction and division (linear coordinates), module and \( \tan^{-1} \) (circular coordinates), hyperbolic module and \( \tanh^{-1} \) (hyperbolic coordinates) functions.

The most efficient way of implementing the CORDIC algorithm is the pipelined one, where each microrotation (Eq. (1)) is implemented in a different stage of the pipeline. This way, all the stages of the pipeline have the same complexity. The shift operations to be performed are known a priori for each stage, avoiding the need for variable length shift registers. The implementation of a radix 2 CORDIC processor needs, for a precision of \( n \) bits, at least \( n \) microrotation stages and approximately \( n/4 \) scale factor compensation stages [5]. If the processor is of a general type, that is, operates in the three coordinate systems, it will be necessary to add stages in order to insure the convergence in hyperbolic coordinates.

### 3. Generalization of the CORDIC algorithm

In the radix \( r \) CORDIC algorithm (where \( r \) is a power of two) the microrotation angles have the value \( \alpha_{i,m} = m^{-1/2} \tan^{-1} (m^{1/2} r^{-i}) \) and \( \sigma_i \in \{-r/2, \ldots, 0, \ldots, r/2\} \). Among these coefficients we include zero, resulting in a redundant representation. For a given precision, the increase of the radix makes the number of microrotations decrease with respect to radix 2. However, it is necessary to determine which is the radix that minimizes the hardware. When using radix different from 2, the coefficients \( \sigma_i \) are different from one and the approximation \( m^{-1/2} \tan(m^{1/2} \pi \sigma_{i,m}) = \sigma_i r^{-i} \) is not valid. In order to obtain the general expression of the
Table 1  
Decomposition of the radix 4 coefficient

<table>
<thead>
<tr>
<th>$\sigma_i$</th>
<th>$\sigma_{i,1}$</th>
<th>$\sigma_{i,2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>$\pm 1$</td>
<td>$\pm 1$</td>
<td>0</td>
</tr>
<tr>
<td>$\pm 2$</td>
<td>$\pm 1$</td>
<td>$\pm 1$</td>
</tr>
</tbody>
</table>

Radix $r$ CORDIC rotations, we will first deduce the radix 4 equations from the radix 2 equations. In an iterative manner we will obtain the CORDIC equations for higher radix.

3.1. Radix 4 CORDIC equations

In radix 4 $\sigma_i \in \{\pm 2, \pm 1, 0\}$. This means that for a given microrotation angle a double rotation, a simple rotation or no rotation can be carried out. By means of the decomposition of the radix 4 microrotation into two successive radix 2 microrotations with the same microrotation angle we can obtain the equations of the radix 4 microrotation. In order to do this, we will express the radix 4 coefficient as the sum of two radix 2 redundant decomposition coefficients (zero is included). The relationship between the radix 4 coefficient ($\sigma_i$) and the radix 2 coefficients ($\sigma_{i,1}$ and $\sigma_{i,2}$) is $\sigma_i = \sigma_{i,1} + \sigma_{i,2}$ (see Table 1). This relationship means that we understand a radix 4 microrotation as the application of two radix 2 microrotations, with the option of carrying them out or not. The radix 4 CORDIC equations obtained are

\[
x_{i+1} = x_i + m(\sigma_{i,1} + \sigma_{i,2})4^{-i}y_i - m\sigma_{i,1}\sigma_{i,2}4^{-2i}x_i
\]
\[
y_{i+1} = y_i - (\sigma_{i,1} + \sigma_{i,2})4^{-i}x_i - m\sigma_{i,1}\sigma_{i,2}4^{-2i}y_i
\]
\[
z_{i+1} = z_i - (\sigma_{i,1} + \sigma_{i,2})\alpha_{i,m}.
\]

With the radix 4 CORDIC equations the convergence is insured despite halving the number of microrotations. The scale factor introduced in each of the microrotations is given by $k_i = (1 + m |\sigma_{i,1}|4^{-2i})^{1/2} \cdot (1 + m |\sigma_{i,2}|4^{-2i})^{1/2}$. The final scale factor is given by

\[
S \frac{1}{2} = \prod_{i=0}^{S-1} (1 + m |\sigma_{i,1}|4^{-2i})^{1/2} \cdot (1 + m |\sigma_{i,2}|4^{-2i})^{1/2},
\]

where $S$ is the number of radix 4 stages. $K$ is not a constant with respect to the decomposition coefficients ($\sigma_{i,2}$ can take the value zero) and it depends on the initial data.

3.2. Radix r CORDIC equations

In order to obtain the radix $r$ CORDIC equations we apply the radix $r/2$ CORDIC equations twice, expressing the radix $r$ decomposition coefficients as the sum of two radix $r/2$ coefficients. Continuing in this manner, we can obtain the general expression for the radix $r$ CORDIC equations

\[
c_{i+1}^k = c_i^k + \sum_{j=1}^{j=r/2} [\text{sign}(\sigma_i)]^j F(k, j) m^{\alpha(k,j)} \binom{\sigma_i}{j} r^{-i} c_i^{e(k,j)}
\]
with \( i = 0, 1, \ldots, S - 1 \), and \( k = 0, 1 \); where \( S \) is the number of microrotations in radix \( r \). Also \( c_i^0 = x_i \) and \( c_i^1 = y_i \), and where

\[
e(k, j) = \frac{1}{2} \left[ 1 - (-1)^{(j+k)} \right]
\]

\[
F(k, j) = (-1)^{(j/2) + j \cdot k}
\]

\[
P(j, k) = 1 - k + \frac{(j - 1)}{2}.
\]

In Eq. (6) we encompass the two CORDIC equations corresponding to the \( x \) and \( y \) coordinates. Each one of them is a polynomial of degree \( r/2 \) in the term \( r^{-i} \). The final scale factor is given by the following expression

\[
K = \prod_{i=0}^{\frac{S-1}{2}} \prod_{p=1}^{\frac{r}{2}} \left[ 1 + m_i \cdot \sigma_{i,p} \cdot r^{-2i} \right]^{1/2}
\]

where \( \sigma_{i,p} \in \{-1, 0, 1\} \) and this coefficients satisfy \( \sum_{p=1}^{r/2} \sigma_{i,p} = 1 \).

In order to obtain the number of microrotations, \( S \), necessary for each radix we must take into account that the number of iterations is given by the precision of the algorithm and it will be determined by the longest of the right shifts in the CORDIC equations (\( n - 1 \) for \( n \) bit precision) and that the decomposition coefficients have a maximum value of \( r/2 \). Then we can deduce the number of microrotations

\[
S = \left\lceil \frac{n - 1 + \log_2(r/2)}{\log_2(r)} \right\rceil
\]

Going from radix 2 to radix 4 means halving of the number of microrotations. Further increases in the radix give a lesser reduction, and the number of additions increases.

3.3. Comparative analysis

In the pipelined implementation of the CORDIC algorithm, the most costly element in silicon area is the adder, therefore, the number of additions determines in a reliable manner the hardware complexity of the processor. In this analysis we must take into account the additions due to Eq. (6) and additions due to the multiplication by coefficients which are not integer powers of two. These coefficients correspond to a combinatory number and this forces the introduction of further additions in order to carry out the multiplication. However, in a pipelined design, throughout the pipeline there will be terms which exceed the precision of the algorithm, as shifts larger than the word size are performed. This is reflected in the fact that in the iteration being evaluated, part of the addition operations of Eq. (6) do not have to be carried out. Therefore, we must calculate an effective number of the additions mentioned.

In Fig. 1 we represent the number of effective additions versus the radix for two typical precisions, 16 and 24 bits, and for the three coordinate systems. In this analysis we have not taken into account the \( z \) coordinate, neither the compensation of the scale factor. For radix 4 a minimum is obtained, with savings of 25% in additions with respect to radix 2 in circular and hyperbolic coordinates and of 50% in the case of linear coordinates. Further increases in the radix make the number of additions grow, due to the multiplications times the coefficients. From the results obtained we can deduce that radix 4 is the optimum one as it minimizes the number of microrotations and the number of adders.
4. Radix 4 CORDIC processor

In this section we will present the radix 4 CORDIC algorithm for the two modes of operation, vectorization and rotation. In the radix 2 CORDIC algorithm, the criterion for the determination of the sense of the microrotation is simple, as it is determined from the sign of \( z_i \) (rotation) or \( y_i \) (vectorization). In radix 4 this criterion is not enough, because apart from determining the sense of the microrotation, we must determine the multiplication factor 0, 1 or 2 associated with it. The next two lemmas present a criterion for both operation modes, for the radix 4 CORDIC algorithm. They will be demonstrated in Appendix A.

**Lemma 1.** The convergence of the radix 4 CORDIC algorithm in the rotation mode is insured if we take

\[
\sigma_i = \begin{cases} 
0 & \text{if } |z_i| \leq \frac{\beta_i}{2} \\
\text{sign}(z_i) & \frac{\beta_i}{2} \leq |z_i| < \frac{3\beta_i}{2} \\
2 \cdot \text{sign}(z_i) & |z_i| \geq \frac{3\beta_i}{2}
\end{cases}
\]

where \( \beta_0 = 2^{-1} \) and \( \beta_i = 4^{-i} \) for \( 1 \leq i \leq S - 1 \).

**Lemma 2.** The convergence of the radix 4 CORDIC algorithm in the vectorization mode is insured if we take

\[
\sigma_i = \begin{cases} 
0 & \text{if } |y_i| \leq \frac{\beta_i \cdot x_i}{2} \\
\text{sign}(y_i) & \frac{\beta_i \cdot x_i}{2} \leq |y_i| < \frac{3 \cdot \beta_i \cdot x_i}{2} \\
2 \cdot \text{sign}(y_i) & |y_i| \geq \frac{3 \cdot \beta_i \cdot x_i}{2}
\end{cases}
\]

where \( \beta_0 = 2^{-1} \) and \( \beta_i = 4^{-i} \) for \( 1 \leq i \leq S - 1 \).
Table 2
Microrotation angles for the three coordinate systems

<table>
<thead>
<tr>
<th>Microrotation</th>
<th>Angle (Circular)</th>
<th>Angle (Hyperbolic)</th>
<th>Angle (Linear)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.463647609</td>
<td>0.549306144</td>
<td>0.500000000</td>
</tr>
<tr>
<td>1</td>
<td>0.244978663</td>
<td>0.255412811</td>
<td>0.250000000</td>
</tr>
<tr>
<td>2</td>
<td>0.062418810</td>
<td>0.062581571</td>
<td>0.062500000</td>
</tr>
<tr>
<td>3</td>
<td>0.015623729</td>
<td>0.015626271</td>
<td>0.001562500</td>
</tr>
<tr>
<td>4</td>
<td>0.003906230</td>
<td>0.003906269</td>
<td>0.003906250</td>
</tr>
<tr>
<td>5</td>
<td>0.000976562</td>
<td>0.000976562</td>
<td>0.000976562</td>
</tr>
<tr>
<td>6</td>
<td>0.000244140</td>
<td>0.000244140</td>
<td>0.000244140</td>
</tr>
<tr>
<td>7</td>
<td>0.000061035</td>
<td>0.000061035</td>
<td>0.000061035</td>
</tr>
<tr>
<td>8</td>
<td>0.000015258</td>
<td>0.000015258</td>
<td>0.000015258</td>
</tr>
<tr>
<td>9</td>
<td>0.000003814</td>
<td>0.000003814</td>
<td>0.000003814</td>
</tr>
<tr>
<td>10</td>
<td>0.000000953</td>
<td>0.000000953</td>
<td>0.000000953</td>
</tr>
<tr>
<td>11</td>
<td>0.000000238</td>
<td>0.000000238</td>
<td>0.000000238</td>
</tr>
</tbody>
</table>

Also, with this criterion, in hyperbolic coordinates it is not necessary to introduce repetition of microrotations in order to insure the convergence. This way, the number of stages of the processor is smaller. In Table 2 we present the final microrotation angles for the three coordinate systems. We have selected \( \tan^{-1}(2^{-1}) \) as the first microrotation angle in circular coordinates, instead \( \tan^{-1}(4^{-0}) \), in order to insure the convergence with the criterion for radix 4 rotation mode expressed in Lemma 1. For the rotation mode, the comparison points are known a priori, and they are simple powers of two. In this way, this criterion can be easily implemented. However, for the vectorization mode, the comparison points are not known a priori (they depend on the value of the \( x \) coordinate in each stage) the comparator can be complex, with respect to the implementation of the criterion for radix 2 vectorization mode. For this reason we will only consider the rotation mode.

4.1. Implementation of the rotation criterion

In order to reduce the delay in each stage, we will include signed digit redundant arithmetic in our design [4,16]. This way we parallelize the adders and reduce the delay making it independent from the precision. Expression (10) is a generic formulation of the rotation criterion for the radix 4 CORDIC algorithm. In this section we are going to particularize this criterion for the case of radix 2 signed digit redundant arithmetic. A number \( x \) is represented as \((x_{-p}, \ldots, x_0, x_1, \ldots, x_n)\) where \( x_i \) belongs to the set \( \{1, 0, -1\} \). In the first place we must study the limits of the \( z \) coordinate as the microrotations are produced. The convergence condition for the radix 4 CORDIC algorithm in iteration \( i \) for any of the coordinate systems is given by expression

\[
| z_i | \leq 2 \cdot \sum_{j=i}^{n/2-1} a_{j,m} < 2^{-2i+2}. 
\]
In iteration \(i\), the most significant digit of the \(z\) coordinate will be at most in the \(2i - 1\) position. On the other hand, from the criterium obtained in expression (10), the coefficient \(\sigma_i\) is zero if \(|z| < 2^{-(2i+3)/2} = 4^{-(i+1)/2}\). In this case the most significant digit of the \(z\) coordinate in iteration \(i\) is at most in the \(2i + 2\) position. On the other hand, in iteration \(i\) we have to estimate the value of the \(z\) coordinate with a precision of \(2^{-(2i+3)}\), which is sufficient in order to guarantee the convergence. Therefore, we only need to evaluate five digits in each iteration. That is, we must consider the digits \(2i - 1, 2i, 2i + 1, 2i + 2\) and \(2i + 3\). We define the truncation operator, \(\tau(\cdot)\), over a number \(x\) in the radix 2 signed digit fractional representation as

\[
\tau(x_{-p} \ldots x_0, x_1 \ldots x_n) = x_{-p} \ldots x_0.
\]

(13)

Taking into account expression (12), it is easy to prove that by shifting the value of the \(z\) coordinate in iteration \(i\), \(2i + 3\) positions to the left, and cutting off its value it verifies

\[
\tau(2^{2i+3} \cdot |z|) \leq 2^{3} \cdot 3.
\]

(14)

As the most significant digit of the \(z\) coordinate in iteration \(i\) is \(2i - 1\), we can take as the cut off value of the \(z\) coordinate a five digit integer as follows

\[
\tau(2^{2i+3} \cdot |z|) \equiv [z_{2i-1}, z_{2i}, z_{2i+1}, z_{2i+2}, z_{2i+3}].
\]

(15)

This integer, due to expression (14) can only take values \(-24, -23, \ldots, -2, -1, 0, 1, \ldots, 23, 24\). Therefore, by testing the value of these five digits in each stage and comparing it to the values of this set we can determine the value of \(\sigma_i\). In order to determine the sign of the coefficient \(\sigma_i\), it is sufficient to detect the sign of the five digits of the \(z\) coordinate we consider in each iteration. For calculating the magnitude of \(\sigma_i\) we check the value of the integer expressed in (15). In this way, the criterion for radix 4 rotation mode in signed digit arithmetic is

\[
\sigma_i = \begin{cases} 
0 & \text{if} \quad -3 \leq [z_{2i-1}, z_{2i}, z_{2i+1}, z_{2i+2}, z_{2i+3}] \leq 3 \\
1 & \text{if} \quad 4 \leq [z_{2i-1}, z_{2i}, z_{2i+1}, z_{2i+2}, z_{2i+3}] \leq 11 \\
-1 & \text{if} \quad -11 \leq [z_{2i-1}, z_{2i}, z_{2i+1}, z_{2i+2}, z_{2i+3}] \leq -4 \\
2 & \text{if} \quad 12 \leq [z_{2i-1}, z_{2i}, z_{2i+1}, z_{2i+2}, z_{2i+3}] \leq 24 \\
-2 & \text{if} \quad -24 \leq [z_{2i-1}, z_{2i}, z_{2i+1}, z_{2i+2}, z_{2i+3}] \leq -12.
\end{cases}
\]

(16)

Figure 2 shows the structure corresponding to a stage of the \(z\) coordinate where we include the criterium for the rotation mode. The SIGN block is a combinational circuit which detects the sign of the integer number made up by the five most significant digits of the \(z\) coordinate. The MAGNIT block is a combinational circuit that calculates the magnitude of the \(\sigma_i\) coefficient. The logic operation of this circuit is shown in Table 3. The current value of the \(z\) coordinate, \(z_{i+1}\), is obtained from an addition/subtraction of \(z_i\) and the microrotation angle obtained from the 3:1 multiplexors. In the addition, one of the operands (the microrotation angle) is expressed in two's complement and the other in radix 2 signed digit. With this, the complexity of the resulting adder is similar to that of the carry ripple [18]. As we show in Fig. 2, in the output of the adder we must introduce another block, we call it COMPACT, whose mission is to insure the condition given in expression (12), with respect to the most significant digit of \(z\) in each microrotation. In the addition/subtraction operation over the \(z\) coordinate in iteration \(i - 1\) there can be significant digits up to position \(2i - 3\), due to the redundant representation used [17]. If we consider the set of digits \(2i - 3, 2i - 2\) and \(2i - 1\) as integer, we know by condition (12) that their value belongs to the set \(-1,0,1\). Therefore, the block COMPACT eliminates this redundancy.

The implementation of the \(x\) and \(y\) coordinates need 4 adders, two for each of the coordinates (see Fig. 3). In each of these stages we have two rows of 2:1 multiplexors, where
we carry out the multiplication of the coefficients times the shifted value of the coordinate, which corresponds to the third term in Eq. (4). In the two rows of 3:1 multiplexors we perform a similar multiplication, but corresponding to the second term of Eq. (4). However, in the last $n/4$ stages of the processor, the complexity of the microrotation is reduced. The two higher adders and their associated multiplexors disappear as one of terms of these adders exceeds the precision of the algorithm and is therefore considered null. The adders used in the implementation of the $x$ and $y$ coordinates are signed digit [21]. The total number of microrotations of the algorithm is $n/2$, with the structure shown in Figs. 2 and 3. The number of adders is $n/2$ for the $z$ coordinate and $2n - n/2$ for the $x$ and $y$ coordinates. This architecture is the same for the three coordinate systems.

4.2. Compensation of the scale factor

In the radix 4 CORDIC algorithm, the scale factor is not constant with respect to the decomposition coefficients (for circular and hyperbolic coordinates) as they have values in $\{0, \pm 1, \pm 2\}$, and this is the main disadvantage of radix 4 respect to radix 2. In this section we

Table 3
Logic operation of the MAGNIT circuit

| a | b | $|z| < 4^{-i}/2$ | $|z| \geq 3 \cdot 4^{-i}/2$ |
|---|---|-----------------|-----------------|
| 0 | 0 | +1              | -1              |
| 0 | 1 | $4^{-i}/2 \leq |z| < 3 \cdot 4^{-i}/2$ | ±1              | 0 |
| 1 | 0 | $|z| \geq 3 \cdot 4^{-i}/2$ | ±1              | ±1 |
introduce a new technique for the compensation of a non constant scale factor, the multifactor compensation with scaling. This technique is based on the calculation of all the possible scale factors. The compensation is obtained by scaling (see Eq. (3)) for each of these factors. The general architecture of a multifactor compensation stage (for a single coordinate) is shown in Fig. 4. It consists of an adder and \( k + 1 \) 2:1 multiplexor stages (where \( k = \lfloor \log_2(n) \rfloor \), being \( n \) the precision of the algorithm). In these multiplexor stages, the wired shifts corresponding to each multifactor compensation stage are carried out, and accepting the possibility that the stage does not contribute to the scaling (adds the value zero). This way, with \( k + 1 \) bits we can control the shift to be performed in each stage or add the value zero. The complete architecture is made up of \( q \) stages similar to the one shown in Fig. 4 (where \( q \) is the maximum number of scaling stages needed by the scale factors to be compensated), and by a control ROM memory, which contains the signals which program the shifts in the 2:1 multiplexors and control the adders in each stage. The control ROM memory has as many words as possible scale factors to be compensated. This word size is estimated at a maximum value of \( (k + 2) \cdot q \).

In Fig. 5 we present the structure of the CORDIC processor we propose. The first microrotations are of double complexity (from \( i = 0 \) to \( i = n/4 - 1 \)), followed by the microrotations of simple complexity (from \( i = n/4 \) to \( i = n/2 - 1 \)). The last stages correspond to the variable scale factor compensation. In Table 4 we summarize the features of the radix 4 CORDIC processor proposed for 16 and 24 bit precision using different coordinate systems. In the first column we specify the number of adders, in the stages column we specify the required number of stages for a radix 2 CORDIC processor \([11]\), the number of effective stages for the radix 4 CORDIC processor (a double complexity stage is considered as two effective stages), and the number of stages corresponding to the multifactor compensation with scaling (stages necessary in a radix 2 CORDIC processor/effective stages of a radix 4 CORDIC processor/stages of multifactor compensation with scaling). In the last column we specify the necessary ROM memory. In order to compare our design with other recent
designs, we have include one more microrotation to obtain the same angle resolution than in these designs.

For 16 bit precision, using only circular coordinate system, the number of possible scale
Table 4
Analysis of the CORDIC processor proposed

<table>
<thead>
<tr>
<th>Coordinate System</th>
<th>Adders</th>
<th>Stages</th>
<th>ROM Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circular 16</td>
<td>49</td>
<td>22/20/7</td>
<td>9x16 + 16x42</td>
</tr>
<tr>
<td>Circular 24</td>
<td>73</td>
<td>31/30/11</td>
<td>13x24 + 64x77</td>
</tr>
<tr>
<td>Circular 16 &amp; Hyperb. 24</td>
<td>49</td>
<td>26/20/7</td>
<td>13x16 + 32x42</td>
</tr>
<tr>
<td>Hyperb. &amp; Linear 24</td>
<td>73</td>
<td>37/30/11</td>
<td>19x24 + 128x77</td>
</tr>
</tbody>
</table>

factors is 16, and the number of scaling stages similar to the one in Fig. 4 is 7, with 5 rows of 2:1 multiplexors per stage. The control ROM memory has 16 x 42 bit words. In this case the radix 4 CORDIC processor (rotation mode) consists in 9 stages for performing the microrotations, but in 4 of these stages, the complexity of the hardware associated with the x and y coordinates is double (4 adders per stage), and these 9 stages are equivalent to 13 effective stages with simple complexity (two adders). However, the z coordinate is implemented in only 9 of these stages. The processor has 20 effective stages, including microrotation and scale factor compensation stages. For a 16 bit precision and using the circular and hyperbolic coordinate system the number of possible scale factors is 32. The number of stages is equal to the circular case and the control ROM memory has 32 x 42 bit words. For 24 bit precision using only circular coordinates, the multifactor compensation has 11 stages similar to the one in Fig. 4, with 6 rows of 2:1 multiplexors. The control ROM memory has 64 x 77 bit words. There are 19 effective stages for microrotations, although the z coordinate is only implemented in 13 of them. In this case, the processor has 30 effective stages. For 24 bit precision using circular and hyperbolic coordinates the number of stages is equal to the circular case and the control ROM memory has 128 x 77 bit words.

5. Evaluation

In this section we will evaluate the radix 4 CORDIC processor we propose, comparing it with other recent CORDIC processors at the level of hardware components. The designs we have used in order to establish the comparison are those of Lange et al. [12], Metafas and Goutis [14], König and Böhme [11] and Takagi et al. [17]. These designs share common characteristics with the one we propose in this paper, such as pipelining and use of scaling for the compensation of the scale factor. In this comparison, we do not take into account an initial stage for the extension of the convergence range or the control structure of each microrotation. In Table 5 we compare our design (radix 4) with the ones we have mentioned. This table encompasses designs for 16 and 24 bit precisions, for circular and hyperbolic coordinate systems or for only the circular coordinate system. For each design we show the latency, effective microrotation stages corresponding to the x and y coordinates, stages which implement the z coordinate, scaling stages, ROM memory, type of arithmetic used, approximate number of adders, and approximate number of multiplexors. In the designs which use signed digit arithmetic (Takagi et al. [17] and Radix 4) we have counted an extra stage
Table 5
Comparative analysis of the different CORDIC processors

<table>
<thead>
<tr>
<th>CORDIC Processor</th>
<th>Coordinate System</th>
<th>Precision</th>
<th>Latency</th>
<th>x/y Stages</th>
<th>z Stages</th>
<th>Scaling Stages</th>
<th>ROM</th>
<th>Arithmetic</th>
<th>Address</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lange [12]</td>
<td>Cir &amp; Hyp.</td>
<td>16</td>
<td>26</td>
<td>26</td>
<td>19</td>
<td>-</td>
<td>26x16 binary</td>
<td>71</td>
<td>168 (2:1)</td>
<td></td>
</tr>
<tr>
<td>Metafas [14]</td>
<td>Circular</td>
<td>16</td>
<td>25</td>
<td>16</td>
<td>16</td>
<td>9</td>
<td>10x16 binary</td>
<td>66</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cir &amp; Hyp.</td>
<td>bits</td>
<td>31</td>
<td>21</td>
<td>21</td>
<td>10</td>
<td>37x16 binary</td>
<td>83</td>
<td>816 (2:1)</td>
<td></td>
</tr>
<tr>
<td>König [11]</td>
<td>Circular</td>
<td>16</td>
<td>22</td>
<td>17</td>
<td>17</td>
<td>5</td>
<td>17x16 binary</td>
<td>61</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>31</td>
<td>28</td>
<td>28</td>
<td>3</td>
<td>28x24</td>
<td>90</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Circular</td>
<td>16</td>
<td>26</td>
<td>21</td>
<td>21</td>
<td>5</td>
<td>30x16</td>
<td>73</td>
<td>176 (2:1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&amp; Hyperb.</td>
<td>24</td>
<td>37</td>
<td>29</td>
<td>29</td>
<td>8</td>
<td>44x24</td>
<td>103</td>
<td>456 (2:1)</td>
<td></td>
</tr>
<tr>
<td>Takagi [17]</td>
<td>Circular</td>
<td>16</td>
<td>24</td>
<td>18</td>
<td>18</td>
<td>9</td>
<td>18x16 signed digit</td>
<td>66</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>37</td>
<td>27</td>
<td>27</td>
<td>9</td>
<td>27x24</td>
<td>101</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Radix 4</td>
<td>Circular</td>
<td>16</td>
<td>21</td>
<td>13</td>
<td>13</td>
<td>7</td>
<td>9x16 + 10x42 signed digit</td>
<td>51</td>
<td>1248 (2:1) 412 (3:1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>31</td>
<td>19</td>
<td>13</td>
<td>11</td>
<td>13x24 + 64x77</td>
<td>75</td>
<td>3456 (2:1) 936 (3:1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Circular</td>
<td>16</td>
<td>21</td>
<td>13</td>
<td>13</td>
<td>7</td>
<td>13x16 + 32x42</td>
<td>51</td>
<td>1392 (2:1) 432 (3:1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&amp; Hyperb.</td>
<td>24</td>
<td>31</td>
<td>19</td>
<td>13</td>
<td>11</td>
<td>19x24 + 128x77</td>
<td>75</td>
<td>3768 (2:1) 936 (3:1)</td>
<td></td>
</tr>
</tbody>
</table>

corresponding to the conversion between redundant arithmetic and two's complement binary arithmetic.

The design of Lange et al. [12] is for 16 bit precision and for hyperbolic and circular coordinates. It introduces redundant microrotations with double shifts in order to force the final scale factor into a power of two, making it very easy to compensate. It is made up of 19 stages, but the first 7 present double shift, and therefore are of double complexity. The number of effective stages is 26, taking into account that only 19 of them implement the z coordinate. The approximate ROM memory it requires is 26 × 16 hit words for the storage of the microrotation angles. Its main feature is that all of the functions (circular and hyperbolic) are calculated in the same computation time, and the range of convergence is the same for the two coordinate systems.

The design proposed by Metafas and Goutis [14] is for 16 bit precision. In this design two algorithms are integrated, the CORDIC algorithm and the CCM algorithms [6]. Initially two types of stages are proposed, both with double complexity, although in one of them the z coordinate is implemented in a simple manner. For establishing a comparison with our design we have only taken into account the stages which are strictly necessary in order to implement the CORDIC algorithm, as well as a particularization of the hardware of each stage. For circular and hyperbolic coordinates, the number of total effective stages is 31, 21 microrotations, 10 scaling stages, and a ROM memory with 37 × 16 bit words. In the case of only circular coordinates, we have particularized the hardware of its initial design and estimated that the total number of stages is 25, with 16 microrotations, 9 scaling stages, and a ROM memory of 16 × 16 bit words.

König and Böhme [11] present several designs, for 16 and 24 bit precision, both, for circular coordinates and for joint circular and hyperbolic coordinates. In their designs they
avoid double shifts and compensate the scale factor with scaling and repetitions. By means of certain convergence restrictions they try to find a set of coefficients which minimize the total number of stages. For 16 bit precision in circular and hyperbolic coordinates it has a total of 26 stages, 21 microrotations, 5 scaling stages, and a ROM memory with $30 \times 16$ bit words. In the case of only using circular coordinates, the total number of stages is 22, $(17 + 5)$ and a ROM memory with $17 \times 16$ bit words. For 24 bit precision in circular and hyperbolic coordinates, the total number of stages is 37, $(29 + 8)$ and a ROM memory with $44 \times 24$ bit words. For 24 bit precision in circular and hyperbolic coordinates, the total number of stages is 37, $(29 + 8)$ and a ROM memory with $44 \times 24$ bit words. For the case of only circular coordinates, the number of total stages is 31 $(28 + 3)$.

Takagi et al. [17] propose a CORDIC rotator in circular coordinates, for the calculation of sine and cosine trigonometric functions. It uses radix 2 signed digit redundant arithmetic like our design. In order to insure a constant scale factor, they introduce two techniques, double rotation method and the correction rotation method. The double rotation method is based on performing each microrotation as two successive microrotations of half the angle with coefficients in the set $\{-1, +1\}$. This way, in all the microrotations (double) a scale factor is introduced independently from the fact that the microrotation is carried out or not. This way, they manage to make the scale factor constant. The correction rotation method consists in only using the values $+1$ and $-1$ for the coefficient $\sigma_i$ despite using redundant arithmetic. This causes that after a certain number of microrotations, there has to be an extra correction microrotation in order to insure the convergence. This technique guarantees a constant scale factor. The optimal solution is obtained by combining both techniques. Thus they propose the use of the correction rotation method in the first half of the stages and the double rotation method in the second half. The algorithm they initially propose is for the calculation of the trigonometric functions and the scale factor is compensated by introducing the data scaled. Another aspect to be taken into account is that the range of the input angles is limited to $-\pi/4 < \theta < \pi/4$. In order to establish a comparison with this design and the one propose in this paper, we are going to consider it as a rotator in circular coordinates. This means that the scale factor must be compensated with the usual techniques for the compensation.

The use of signed digit arithmetic in the radix 4 CORDIC processor proposed in this paper permits the parallelization of the operation of the adders, reducing the delay of each microrotation with respect to implementations which use non redundant binary arithmetic. By means of the introduction of radix 4 in all the CORDIC microrotations we significantly reduce the number of adders and the latency (effective stages) with respect to the other implementations in the table. This difference is more pronounced when circular and hyperbolic coordinates are used together, as in radix 2 solutions the use of hyperbolic coordinates makes the introduction of microrotation repetitions necessary in order to insure convergence. On the other hand, the compensation of non-constant scale factor, by means of the technique developed in this work, permits the elimination of the main limitation of the generalized application of radix 4 to the design of high speed CORDIC processors with the cost of increasing the area of ROM memory and multiplexors. Finally, the generalization of the CORDIC algorithm to radix 4, the development of a technique for the compensation of a non constant scale factor, and the use of redundant arithmetic, has allowed us to develop a new general application CORDIC processor architecture which optimizes characteristics of previous designs.

Appendix A

**Lemma 1.** The convergence of the radix 4 CORDIC algorithm in the rotation mode is insured if we select the $\sigma_i$ coefficient with the criterium expressed in (10).
Proof. The convergence condition in the radix 4 CORDIC algorithm is $|z_{n/2}| \leq \Delta z_{m,n}$, with $\Delta z_{m,n} = m^{-1/2} \tan^{-1}(m^{1/2} 2^{-n+1})$, for $n$ bit precision. Let $\alpha_i = m^{-1/2} \tan^{-1}(m^{1/2} \beta_i)$, $C_i,m = 2 \cdot \sum_{i=0}^{n/2-1} \alpha_i,m + \Delta z_{m,n}$, $\beta_0 = 2^{-1}$ and $\beta_i = 4^{-i}$ for $1 \leq i \leq n/2 - 1$. At the first time, we have to demonstrate by induction over $i$ that $|z_{i+1}| \leq C_{i+1,m}$, for $i = 0, \ldots, n/2 - 2$, and then we will demonstrate that $|z_{n/2}| \leq \Delta z_{m,n}$. We have to take into account that $z_{i+1} = z_i - \sigma_i \alpha_{i,m}$ and $|z_0| \leq C_{0,m}$. For $i = 0$, we have three possible different cases

(i) If $|z_0| < \beta_0/2$, then $|\sigma_0| = 0$ and $|z_{i+1}| < \beta_i/2$.
(ii) If $3 \beta_0/2 > |z_0| \geq \beta_0/2$, then $|\sigma_0| = 1$ and $|z_{i+1}| \leq (2m + 1) \beta_0/2 - m \alpha_{0,m} \leq C_{1,m}$.
(iii) If $C_{0,m} \geq |z_0| \geq 3 \beta_0/2$, then $|\sigma_0| = 2$ and $|z_{i+1}| \leq C_{1,m}$.

In all cases the condition $|z_{i+1}| \leq C_{i+1,m}$ is satisfied. Now we suppose that $|z_i| \leq C_{i,m}$, then, as before, we have three possible cases

(i) If $|z_i| < \beta_i/2$, then $|\sigma_i| = 0$ and $|z_{i+1}| < \beta_i/2$.
(ii) If $3 \beta_i/2 > |z_i| \geq \beta_i/2$, then $|\sigma_i| = 1$ and $|z_{i+1}| \leq (2m + 1) \beta_i/2 - m \alpha_{i,m}$.
(iii) If $C_{i,m} \geq |z_i| \geq 3 \beta_i/2$, then $|\sigma_i| = 2$ and $|z_{i+1}| \leq C_{i+1,m}$.

If we take into account that $\beta_i/2 \leq (2m + 1) \beta_i/2 - m \alpha_{i,m} \leq C_{i+1,m}$, then in all of the cases $|z_{i+1}| \leq C_{i+1,m}$. In order to obtain the remainder angle we taken $i = n/2 - 1$ (last radix 4 microrotation). In this particular case $|z_{n/2-1}| \leq C_{n/2-1,m} = 2^2 \cdot m^{2-n+1}$ and $\Delta z_{m,n}$. On the other hand we can express $m^{-1/2} \tan^{-1}(m^{1/2} 2^{-n+2}) = 2^{-n+1} \cdot m^{2-n+1}/3 + \ldots$, for $i > n/2 - 2$, we can approximate $m^{-1/2} \tan^{-1}(m^{1/2} 2^{-2i}) = 2^{-2i}$ for $n$ bit precision. Again we have three possible cases:

(i) If $|z_{n/2-1}| < 2^{-n+1}/2$, then $|\sigma_{n/2-1}| = 0$ and $|z_{n/2}| < 2^{-n+1} = \Delta z_{m,n}$.
(ii) If $3 \cdot 2^{-n+1}/2 > |z_{n/2-1}| \geq 2^{-n+1}/2$, then $|\sigma_{n/2-1}| = 1$ and $|z_{n/2}| < 2^{-n+1} = \Delta z_{m,n}$.
(iii) If $C_{n/2-1,m} \geq |z_{n/2-1}| \geq 3 \cdot 2^{-n+2}/2$, then $|\sigma_{n/2-1}| = 2$ and $|z_{n/2}| \leq \Delta z_{m,n}$.

For all of the cases $|z_{n/2}| \leq \Delta z_{m,n}$, and the convergence is insured.

Lemma 2. The convergence of the radix 4 CORDIC algorithm in the vectorization mode is insured if we select the $\sigma_i$ coefficient with the criterium expressed in (11).

Proof. In radix 4 the convergence condition can be expressed as $\tan^{-1}(y_{n/2}/x_{n/2}) \leq m^{-1/2} \tan^{-1}(m^{1/2} 2^{-n+1})$. For the vectorization mode, we can transform the $y$ coordinate equation as $y_{i+1} = [(y_i/x_i) \cdot (1 - m \sigma_i (\sigma_{i+1} 2^{i})) - (\sigma_{i+1} + \sigma_i) \beta_i] \cdot x_i$. If we identify $y_i/x_i$ in this equation with $z_i$ in the equation corresponding to the rotation mode, and taking into account that $x_i < x_{i+1}$, then we can obtain in a similar way to the previous lemma that $|y_{n/2}| \leq 2^{-n+1} \cdot x_{n/2}$. For the vectorization mode we can impose the condition $|y_i| \leq |x_i| < 1$, and then $|y_{n/2}| \leq 2^{-n+1} \cdot \sqrt{2}$. In this case we can approximate $\tan^{-1}(y_{n/2}/x_{n/2}) = y_{n/2}/x_{n/2} \leq 2^{-n+1} \cdot m^{-1/2} \tan^{-1}(m^{1/2} 2^{-n+1})$, and the lemma is proved.

References


