A FAST ARCHITECTURE FOR RADIX 10 COORDINATES ROTATION

Antonio Jimeno Morenilla, Higinio Mora Mora, Jose-Luis Sanchez Romero, Francisco Pujol Lopez

Computer Science Technology Department
University of Alicante
Apdo. Correos 99. 03080 Alicante, Spain
email: {jimeno, sanchez, hmora, fpujol}@dtic.ua.es

ABSTRACT
Although Radix-10 based arithmetic has been gaining renewed importance over the last few years, decimal systems are not efficient enough and techniques are still under development. In this paper, a modification of the CORDIC method for decimal arithmetic is proposed and applied to produce fast rotations. The algorithm uses BCD operands as inputs, combining the advantages both decimal and binary systems. The result is an important number of iterations reduction compared with the original decimal CORDIC method. Finally, a FPGA-based radix-10 architecture that can be used to produce rotations with more precision and speed is presented and different experiments showing the advantages of the new method are shown.

1. INTRODUCTION
Binary system is the mathematical substrate of digital systems so the use of binary arithmetic is present in most of computation systems. On the other hand, decimal is easier for humans to handle mainly because we have ten fingers and we are just used to thinking that way. In fact, first computing machines worked with decimal representations [1]. The binary number system offers many advantages over a decimal representation for a high-performance, general purpose computer. The greater simplicity of a binary arithmetic unit and the greater compactness of binary numbers both contribute directly to arithmetic speed. This fact has finally led to a preponderance of binary systems over decimal ones. Nevertheless, throughout the evolution of computers, some proposals have been given about the need of finding a balance between both binary and decimal systems, since decimal numbers are essential for communicating between man and the computers. For this reason, various computers with decimal calculation have been developed [2]-[7]. Different Hewlett Packard [8], Texas Instruments [9] and Casio calculators also use decimal formats [10].

There are many examples of fields where information on computers is stored under decimal format. For example, in Computer Aided Design (CAD), the use of radix 10 representation and, at the same time, high precision are required. For instance, in the creation phase prior to manufacturing, engineers use the decimal system. When defining radix 10 magnitudes such as 0.1 for an object, the internal use of radix 2 representation usually creates an inherent error, since the equivalent binary numbers will have an infinite amount of fractional digits. Moreover, in Computer Aided Manufacturing (CAM) the use of ASCII for data interchange is also widespread. There were several reasons for this decision. First, humans learn about formats much more efficiently from short example files than pages of formal specifications, and ASCII is human-readable. Humans would much prefer to examine an ASCII file, even if binary versions are available. On the other hand, there are currently optic and magnetic sensors which directly provide the output in BCD format, so that the user can easily monitor the evolution of certain magnitudes and detect any errors [11]. The are other fields like financial and business-oriented applications where studies have revealed that 55% of the numerical data contained in commercial databases are in decimal format [12]. Recently the IBM z900 microprocessor has been developed [13], with a decimal arithmetic unit.

Proof of the importance recently given to decimal representation is the fact that the standard IEEE 854 has been expanded to support decimal floating point [14]. Likewise, the first draft of the new standard floating point representation system proposes, explicitly, different formats of decimal exchange and different types of applications in which decimal formats are especially suitable [15].

Several methods for performing decimal arithmetic operations with binary logic have been proposed, from the most elementary ones, such as addition and product, to other more complex ones, such as division, square root, and logarithm [16]. Other functions, such as the trigonometric ones, can be obtained in a simple, precise and relatively fast way by means of the CORDIC (COordinate Rotation Digital Computer) method [17]. Originally, CORDIC was applied to binary arithmetic, but later its application was proposed for decimal representation of data [19], [20]. In this work, an improvement of decimal CORDIC is proposed to make coordinate rotations.
2. THE CORDIC METHOD

2.1. Reviewing the binary CORDIC Method
CORDIC was developed by Volder [17] for calculating the rotation of a two-dimensional vector with circular coordinates represented in the binary system, exclusively using addition and shift operations. Walther [18] extended the method application to hyperbolic and linear coordinates. The method works in two modes: rotation mode and vectoring mode. In the rotation mode, a vector \((x, y)\) is rotated through an angle \(\theta\) in order to obtain a new vector \((x_n, y_n)\). The overall rotation is divided into micro-rotation so that, in micro-rotation \(j\), an angle \(\alpha_j = \tan^{-1}(2^{-j})\) is added to or subtracted from the remaining angle \(\theta\). In this way, this angle approaches zero. The elementary angles \(\alpha_j\) must fulfill the following condition [17]-[19]:

\[
\alpha_j \leq \sum_{k=j+1}^{n} \alpha_k + \alpha_n, \quad j \geq 0
\]

The algorithm is based on the following equations:

\[
x_{j+1} = x_j - m \sigma_j y_j 2^{-d(j)}
\]
\[
y_{j+1} = y_j + \sigma_j x_j 2^{-d(j)}
\]
\[
z_{j+1} = z_j - w_{d(j)}
\]

The values for \(m\) are 1 for circular, -1 for hyperbolic, and 0 for linear coordinates. The value for \(\sigma\) determines the direction of micro-rotation \(j\). In the rotation mode, \(\sigma\) is equal to 1 if \(z_j\) is positive and is equal to -1 otherwise. The values for \(d(j)\), \(w_{d(j)}\) and \(m\) are shown in Table 1. It shows that, with hyperbolic coordinates, carrying out each micro-rotation only once is not sufficient. Indeed, convergence can be achieved by repeating certain iterations [16].

### Table 1. Parameters for Different Coordinate Systems

<table>
<thead>
<tr>
<th>Coord. type</th>
<th>(d(j))</th>
<th>(w_{d(j)})</th>
<th>(m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circular</td>
<td>(j)</td>
<td>(\tan^{-1}(2^{2j}))</td>
<td>1</td>
</tr>
<tr>
<td>Hyperbolic</td>
<td>(j-k), (k) is the largest integer such that (3^k+2k-1 \leq 2j)</td>
<td>(\tanh^{-1}(2^{2j}))</td>
<td>-1</td>
</tr>
<tr>
<td>Linear</td>
<td>(j)</td>
<td>(2^{2j})</td>
<td>0</td>
</tr>
</tbody>
</table>

With regard to the elementary angles chosen for circular coordinates, convergence is guaranteed since the following property is accomplished:

\[
\tan^{-1}(2^{-j}) \leq \sum_{k=j+1}^{n} \tan^{-1}(2^{-k}), \quad j \geq 0
\]

Table 2 shows the results provided by the algorithm depending on the type of coordinates. As shown in Table II, one of the advantages of the method is the computation of multivariate functions. With slight modifications on the operands, CORDIC can also provide other functions such as exponential, sine and cosine, and so on.

### Table 2. Algorithm Results for Different Coordinates Type

<table>
<thead>
<tr>
<th>Type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circular</td>
<td>(x_n = K_j (x \cos z - y \sin z))</td>
</tr>
<tr>
<td></td>
<td>(y_n = K_j (y \cos z + x \sin z))</td>
</tr>
<tr>
<td></td>
<td>(z_n = 0)</td>
</tr>
<tr>
<td>Hyperbolic</td>
<td>(x_n = K_j (x \cosh z + y \sinh z))</td>
</tr>
<tr>
<td></td>
<td>(y_n = K_j (y \cosh z - x \sinh z))</td>
</tr>
<tr>
<td></td>
<td>(z_n = 0)</td>
</tr>
<tr>
<td>Linear</td>
<td>(x_n = x)</td>
</tr>
<tr>
<td></td>
<td>(y_n = y + x)</td>
</tr>
<tr>
<td></td>
<td>(z_n = 0)</td>
</tr>
</tbody>
</table>

In each iteration \(j\), a scaling is added to the new coordinates \((x_j, y_j)\). The scaling factor in iteration \(j\) is given by the following expression:

\[
K_{m,j} = \sqrt{1 + m 2^{-j}}
\]

Several techniques to avoid performing the final product by \(K_{m,j}^{-1}\) and carry out the compensation of the scaling in parallel with each iteration have been proposed [21]-[23].

2.2. Reviewing the Decimal CORDIC Method
In [16] and [19] the use of CORDIC for decimal operands expressed in BCD representation is proposed. The modification of the method, focusing on the case of circular coordinates, is expressed by the following iterative equations:

\[
x_{j+1} = x_j - \sigma_j y_j 10^j
\]
\[
y_{j+1} = y_j + \sigma_j x_j 10^j
\]
\[
z_{j+1} = z_j - \tan^{-1}(10^j)
\]

The drawback of this modification of the method lies on the relation between any two consecutive elementary angles in the form \(\tan^{-1}(10^j)\). Table 3 shows the values of these angles for radix 2 and radix 10, assuming 10 iterations.

### Table 3. Elementary angles for Radix-2 and Radix-10

<table>
<thead>
<tr>
<th>Stage</th>
<th>(\tan^{-1}(2^j))</th>
<th>(\sum_{k=j}^{n} \tan^{-1}(2^k)) (k &gt; j)</th>
<th>(\tan^{-1}(10^j))</th>
<th>(\sum_{k=j}^{n} \tan^{-1}(10^k)) (k &gt; j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(j = 0)</td>
<td>0.78539816</td>
<td>0.95788846</td>
<td>0.78539816</td>
<td>0.11077943</td>
</tr>
<tr>
<td>(j = 1)</td>
<td>0.46364761</td>
<td>0.49424085</td>
<td>0.09966865</td>
<td>0.01111078</td>
</tr>
<tr>
<td>(j = 2)</td>
<td>0.24497866</td>
<td>0.24926219</td>
<td>0.00999967</td>
<td>0.00111111</td>
</tr>
<tr>
<td>(j = 3)</td>
<td>0.12345500</td>
<td>0.12490719</td>
<td>0.00000000</td>
<td>0.00001111</td>
</tr>
<tr>
<td>(j = 4)</td>
<td>0.06241881</td>
<td>0.06248838</td>
<td>0.00000000</td>
<td>0.00000011</td>
</tr>
<tr>
<td>(j = 5)</td>
<td>0.03123983</td>
<td>0.03124855</td>
<td>0.00000000</td>
<td>0.00000001</td>
</tr>
<tr>
<td>(j = 6)</td>
<td>0.01562373</td>
<td>0.01562482</td>
<td>0.00000000</td>
<td>0.00000001</td>
</tr>
<tr>
<td>(j = 7)</td>
<td>0.00781234</td>
<td>0.00781248</td>
<td>0.00000000</td>
<td>0.00000001</td>
</tr>
<tr>
<td>(j = 8)</td>
<td>0.00390623</td>
<td>0.00390625</td>
<td>0.00000000</td>
<td>1E-09</td>
</tr>
<tr>
<td>(j = 9)</td>
<td>0.00195312</td>
<td>0.00195312</td>
<td>0.00000000</td>
<td>1E-10</td>
</tr>
</tbody>
</table>
The relation between any two consecutive angles in the form $\tan^{-1}(2^j)$ is approximately 2, or even less. This fact facilitates convergence in binary CORDIC, as expressed in (5). However, in the case of decimal representation, each angle is approximately 10 times smaller than the one immediately preceding it, so convergence of the method cannot be directly guaranteed.

As shown in the rightmost column of Table III, the convergence condition in (1) is not satisfied, since the following property is obtained:

$$\tan^{-1}(10^{-j}) > \sum_{k=j+1}^{n} \tan^{-1}(10^{-k}), j \geq 0 \quad (7)$$

According to decimal representation, the relation between two consecutive angles means that every iteration but the initial one must be repeated 9 times in order to achieve convergence [18]. In this case, the following expression is obtained:

$$\tan^{-1}(10^{-j}) \leq 9 \sum_{k=j+1}^{n} \tan^{-1}(10^{-k}), j \geq 0 \quad (8)$$

The results obtained with the decimal CORDIC proposed in [16] and [19] are valid with regard to precision. However, this method cannot compete with binary CORDIC if the latency of the algorithm is taken as a criterion of comparison, since the binary method requires a smaller number of iterations so as to obtain the same precision. Therefore, the advantages derived from the use of the algorithm with BCD representation would be reduced to omit conversion between BCD and binary representation and, consequently, avoid loss of precision.

### 3. THE NEW DECIMAL CORDIC METHOD

The new proposal for the CORDIC method is the result of combining the performance and precision of binary CORDIC with decimal representation in BCD format. That is, the decomposition of the angle to be rotated in elementary angles that fulfill condition (3) will be used and, as a result, convergence is guaranteed without the need of repeating any stage. However, the product by $2^j$ that appears in (2a) and (2b) can no longer be implemented as a simple shift of bits but rather, when using decimal BCD codification, some kind of correcting operation must be performed instead.

Dividing a number by 2 can be conceived as multiplying the same number by 0.5. Therefore, the division can be carried out by shifting 4 times to the right the different BCD digits and then multiplying the result by 5. In [16] multiplying a BCD number by 5 is described taking into account that $b_{i}[0]$ represents the bit in the position $j$ within the BCD digit $b_i$ after shifting and correction:

1. If $b_{i+1}[0]$ is 0, the new digit $b'_{i}$ is constructed by selecting the 3 most significant bits of $b_i$ and shifting them to the right, and setting a 0 in the most significant position.
2. If $b_{i+1}[0]$ is 1, the new digit $b'_{i}$ is calculated by using the following logical equations:

$$b'^{-1}[0] = \overline{b_{i}[0]} \cdot b'^{-1}[1] = b_{i}[2] \oplus b_{i}[1]$$

$$b'^{-1}[2] = b'^{-1}[3] = b_{i}[3] + b_{i}[2] \cdot b_{i}[1] \quad (9)$$

This proposal permits the number of gates to be reduced to 3 AN gates, 2 OR gates, plus 3 inverters, and the number of gate levels the signal passes through is 3. At first glance, the proposed algorithm behaves better than the one described in [16]. However, the method proposed also requires a multiplexer circuit which, depending on the value of the least significant bit of the BCD digit, selects the bits of the shifted digit or the same bits after the correcting addition.

![Fig. 1. The multiplexers for selecting between $b_i$ and $b'_i$](image)

The configuration for this proposal can be seen in Fig. 1. The control signal $c / \overline{c}$ (correction / no correction) comes from bit $b_{i+1}[0]$. If the multiplexer is not regarded as a
subsequent stage but part of the circuit logic, the following equations are obtained:

\[
\begin{align*}
b_i[0] &= b_i[1] \oplus b_i[0] = b_i[1] \overline{b_i[0]} + b_i[1]b_i[0] \\
b_i[1] &= b_i[2]b_i[1] + b_i[2]b_i[0] + b_i[1]b_i[0] \quad (10) \\
\end{align*}
\]

The total number of gates is 12 AND gates, 6 OR gates, and 4 inverters. The number of gate levels the signal passes through is 5, as in the previous case.

4. A HYBRID CORDIC ARCHITECTURE

There are various alternatives for adding two BCD operands [16]. The operation is more complex than binary addition due to the fact that the carry resulting from the sum of two digits must be propagated to the sum of the following ones. Moreover, the sum of two BCD digits must be corrected if it is greater than the maximum supported value, that is, 9. The correction consists in adding the value 6 to the intermediate result. Subtraction has similar characteristics to addition, since it can be obtained by adding the complement of the second term to the first one. In Fig. 2 a joint architecture for BCD addition and subtraction is shown.

![Fig. 2. BCD adder/subtractor; Z = M ± N. Based on the designs in [18].](image)

The complete architecture for each of the iterations of the proposed CORDIC method is shown in Fig 3. The BCD adder is the same as the one represented in Fig. 2. On the other hand, the \(2^{j-1}\)-divider is the shifter of the number of bits indicated by \(j-1\), and is constructed based on the 1-bit shifter shown in Fig 1. In the first iteration, that is, when \(j = 0\), the control logic selects the bits \(X_i[3]\), \(X_i[2]\) \(X_i[1]\), and \(X_i[0]\); whereas in the following iterations the bits selected are \(b+i[3]\), \(b+i[2]\), \(b+i[1]\), and \(b+i+1[0]\)

5. EXPERIMENTATION

The proposed architecture was implemented using the Xilinx ISE (Integrated Software Environment) 7.1i tool. The FPGA XC3S100E from the Spartan3E family was chosen. With the aim of making a suitable comparison, the architectures corresponding to binary CORDIC and decimal CORDIC proposed in [16] and [19] were also implemented. In all cases, a complete stage of the algorithm was implemented, with the type of adder and shifter being varied according to each method. The global method was implemented on an unfolded architecture. In the case of the hybrid method, the architecture was constructed by means of a 24-stage pipeline, each one according to the diagram shown in Fig. 3. Conventional arithmetic was used and the final stage of scaling factor compensation was omitted.

5.1 Experiments on precision

Different tests were carried out so as to make a complete comparison with regard to precision between decimal CORDIC and the hybrid CORDIC proposed in this work. Some tests were also performed to compare binary CORDIC with the proposed method. Values within the range \([0, 0.999]\) were chosen for the coordinates \((x, y)\), and angles were likewise taken within the range \([0, 0.999]\) radians. All the data were represented in natural BCD with 8 fractional digits. For each type of test, an analysis for the relative error distribution was made, as well as for the maximum relative error.

![Fig. 4. Relative error distribution on rotating 30 random points through each of the angles \((x = \text{decimal CORDIC}; \square = \text{hybrid CORDIC})\).](image)

In the first test, 30 random points were taken and rotated through an initial angle 0.01; this angle was successively augmented with an increment of 0.01 until 0.99 was
reached; after each increase, 30 new random points were rotated. The results of the error distribution are shown in Fig. 4, and the maximum error for each angle is indicated by means of a line. The error distribution for the proposed method is significantly lower than that for decimal CORDIC. In addition, the maximum error for any of the angles considered oscillates for decimal CORDIC, whereas for the hybrid method the maximum error presents much less variability.

In a new test, a set point \((x, y)\) was taken and 30 rotations were made through random angles between 0 and 0.999 radians, while varying the number of BCD digits considered from the input data. The error distribution results are shown in Fig. 5. On considering an operand length up to 8 digits, the result is shown to be somewhat high, but the maximum error obtained with the proposed CORDIC method is never higher than that for decimal CORDIC.

The following experiment was aimed at comparing the number of iterations required in each method so as to achieve suitable precision. A fixed point \((x, y)\) was considered and angles between 0 and 0.999 rd were taken at random and the initial point was rotated through them. The results of the error distribution are shown in Fig 6. There is seen to be a decreasing tendency for both methods as the number of iterations increases. However, the slope is much greater for the hybrid method. For this method, the error stabilizes at 24 iterations whereas for decimal CORDIC, around 60 iterations are required. In addition, the maximum error is always lower for the hybrid method. The number of iterations for the hybrid method cannot be higher than 32 since it is the number of bits that the BCD 8-digit operands have.

Finally, a comparison between decimal CORDIC, binary CORDIC, and the hybrid method, was made for 30 random rotations of a set point through angles between 0 and 0.999 radians, while varying the number of iterations.

The results for the maximum relative error are shown in Fig. 7. There is seen to be a decreasing tendency for every method as the number of iterations increases. However, the error decrement is much slower for decimal CORDIC. The error for both binary CORDIC and the hybrid method are always lower than that for decimal CORDIC. For decimal CORDIC around 60 iterations are required, whereas the number of iterations for the hybrid method cannot be higher than 32 since it is the number of bits that the BCD 8-digit operands have. In fact, just 24 iterations have been considered for the hybrid method, since from this iteration on the maximum error stabilizes. This is due to the fact that some of the 32 bits in the input operands are not useful, since when working with BCD format each 4-bit digit can only take 10 out of 16 possible combinations.

### 5.2. Experiments on latency

The delays obtained, in nanoseconds, are shown in Table 5. The results show that the proposed hybrid CORDIC has a delay per stage 4 times lower than that of decimal CORDIC.
until it is completely processed. With regard to this, Fig. 8 shows the latency obtained as the number of data supplied to the architecture increases. The hybrid CORDIC architecture offers better indices of performance than the decimal architecture, both in delay of only one stage and in latency derived from the data processing throughout the overall set of stages.

| Table 5. Single Stage Delay for Different CORDIC Architectures |
|------------------------|-----------------|-----------------|
| Binary                | Decimal         | Hybrid          |
| 71.51                 | 784.76          | 181.12          |

6. CONCLUSION

In this work, a modification of the CORDIC method which permits working with decimal input data represented in BCD format combined with elementary angles of the type \( \tan^{-1}(2^{-j}) \) has been proposed. In relation to this, it has been necessary to define the product of the decimal data by \( 2^{-j} \). The use of this type of elementary angles improves convergence speed. The tests performed confirm that the proposed CORDIC method requires fewer number of iterations, less than 50% compared with decimal CORDIC, so as to obtain a determined precision. All the tests have shown that the maximum error obtained with any variation of the parameters (number of iterations, angle to be rotated, or number of digits considered) is always lower for the proposed method. On the other hand, with regard to latency, the experiments show that the proposed method has a much lower delay than that for decimal CORDIC, both for one only stage and for the complete set of stages. In addition, the indices obtained are not very far from those obtained by binary CORDIC. Other advantages are the possibility of including scaling factor compensation techniques compatible with the ones proposed for binary CORDIC, as well as the fact that an initial conversion from the decimal input data to radix 2 and the later conversion the other way round are avoided.

Finally, it must be pointed out that the importance of operations with numbers originally represented in decimal format is increasing over the last few years. Since there are current fields in which radix 10 data are required, conversion errors incorporated by digital computers should be avoided. The proposed hybrid method is believed to improve performance on systems which use decimal representation and which require both a high precision and a small latency.

7. REFERENCES