Redundant CORDIC Rotator Based on Parallel Prediction *

Elisardo Antelo, Javier D. Bruguera

Dept. Electrónica y Computación
University of Santiago de Compostela
15706 Santiago de Compostela. SPAIN
e-mail: (elisardo,elbrugue)@usc.es

Julio Villalba and Emilio L. Zapata

Dept. Computer Architecture
University of Málaga
29013 Málaga. SPAIN.
e-mail: (villalba,ezapata)@atc.ctima.uma.es

Abstract

In this work we present a Cordic rotator, using carry-save arithmetic, based on the prediction of all the coefficients into which the rotation angle is decomposed. The prediction algorithm is based on the use of radix-2 microrotations with multiple shifts in the first iterations and the use of a redundant radix-2 and radix-4 representation for the coefficients in the rest of the microrotations. The use of multiple shifts facilitates the prediction of the coefficients in the case of microrotations where $i \leq n/4$, being n the precision of the algorithm, and the use of radix-4 microrotations helps to reduce the total number of iterations. The prediction is carried out using the redundant representation of the z coordinate, without any need for conversions to a non-redundant representation. Finally, we present a VLSI architecture based on this algorithm. As the production of the coefficients is very fast, and they are known before starting each microrotation, the resulting architecture can be highly pipelined and consequently appropriate for applications where high speeds are required.

1 Introduction

The CORDIC algorithm is an iterative technique for performing plane rotations and evaluate different trigonometric, hyperbolic and linear functions [14][15]. Its current applications are in the field of digital signal and image processing, filtering, matrix algebra, simulation and robotics [9]. The algorithm is based on the decomposition of the rotation angle into known elementary angles. The basic iteration or microrotation is

$$\begin{aligned} x_{i+1} &= x_i - m\sigma_i 2^{-i} y_i \\ y_{i+1} &= y_i + \sigma_i 2^{-i} x_i \\ z_{i+1} &= z_i - \sigma_i \alpha_{i,m} \end{aligned} \qquad (1)$$

where $\alpha_{i,m} = m^{-1/2}\tan^{-1}(m^{1/2}2^{-i})$ is the microrotation angle, m specifies the coordinate system ($m = 1$ for circular coordinates, $m = -1$ for hyperbolic coordinates and $m = 0$ for linear coordinates) and $\sigma_i \in \{-1,+1\}$ is the microrotation direction. In the rotation mode, where coordinate z is driven to zero, $\sigma_i = sign(z_i)$, and in the vectoring mode, where y is driven to zero, $\sigma_i = -sign(y_i)\cdot sign(x_i)$. The value of the final coordinates is scaled by the factor $K_m = \prod(1 + m\sigma_i^2 2^{-2i})^{1/2}$. Since $\sigma_i \in \{-1,+1\}$, K_m, is constant and must be compensated. In circular coordinates at least $n+1$ microrotations (n in the case of linear and hyperbolic coordinates) are required for n bit precision. However, in hyperbolic coordinates it is also necessary to repeat the microrotations with shifts $4, 13, \ldots, k, 3k+1, \ldots$ in order to guarantee convergence [15].

The propagation of the carry in the adders limits the speed of the CORDIC processors. Redundant carry-save and signed-digit arithmetic have been used in order to eliminate the carry propagation [4][5][6][12]. However, the determination of σ_i requires the assimilation of the redundant representation of the z or y coordinate, depending on the operation mode. A small number of the most significant bits are assimilated so as not to degrade the speed of redundant arithmetic. Because of this, additional correcting microrotations [10][12] must be introduced in order to correct the possible errors when obtaining σ_i. Alternatively σ_i must be allowed to take values in the redundant set $\{-1,0,1\}$ [6] [12].

Another method for the determination of σ_i in redundant arithmetic is based on the parallel prediction of the $\sigma_i's$ [2][13]. The main advantage of the prediction algorithm is that its implementation in VLSI technology is faster than the other methods. However, and due to the fact that the prediction can only be achieved from $i = 1$ on, the convergence range in circular coordinates is smaller than $[-\pi/2, \pi/2]$. On the other hand, in the z coordinate it is necessary to carry out several conversions from redundant to non-redundant arithmetic. This constrains the efficiency of its implementation in a pipelined architecture. The prediction algorithm has been applied to the rotation mode with carry-save arithmetic in non pipelined unfolded architectures [2] [13].

In this work we present a modification of the prediction algorithm for the rotation mode and carry-save arithmetic which eliminates these drawbacks. We propose a new prediction algorithm based on micro-

*This work was supported in part by the Ministry of Education and Science (CICYT) of Spain under contract TIC92-0942 and Xunta de Galicia under contract XUGA-20606B93

rotations with multiple shifts. We incorporate this method into a general algorithm and architecture. On the other hand we use radix-4 microrotations [3][10] [13] that permit reducing the number of stages of the processor. Finally a complexity analysis and comparison with another implementation is made. We only present the modified prediction algorithm for circular coordinates. In order to make reading this work easier, the microrotation angles for circular coordinates, $\alpha_{i,1}$, are represented as α_i.

2 Prediction Algorithm

The prediction algorithm [2][13] consists in selecting the values of σ_i equal to the digits of the z coordinate. It is however necessary to repeat some microrotations in order to correct the error introduced by selecting the $\sigma_i's$ this way. Using a non-redundant representation with n bit precision, z_j can be expressed as,

$$z_j = \sum_{i=j}^{n} c_i \cdot 2^{-i} = \sum_{i=j}^{n} \sigma_i \cdot \alpha_i \qquad (2)$$

with $c_i \in \{-1,+1\}$ instead of the set $\{0,1\}$ (the conversion between these two sets of digits is very simple [13]). In general, $\alpha_i = \tan^{-1}(2^{-i}) \neq 2^{-i}$, although the difference between 2^{-i} and α_i decreases as the value of i increases. If we select $\sigma_i = c_i$ from $i = j$ to $i = k$ an error is accumulated between iterations j and k, because $\alpha_i \neq 2^{-i}$. In order to insure the convergence of the algorithm it is necessary to repeat microrotation k, which corrects this error. The indices j and k must verify the following relation:

$$\sum_{i=j}^{k} \left|2^{-i} - \alpha_i\right| \leq 2^{-k} \qquad (3)$$

This expression permits determining how many $\sigma_i's$ may be predicted from microrotation j before introducing a correcting microrotation in k. Observe that we have imposed the bound 2^{-k} instead α_k. This bound is less strong and we can use it because a 1 or a -1 are allowed in the position with weight 2^{-k} since we begin the next prediction from the position k. The difference between 2^{-k} and α_k is corrected by the next correcting microrotation.

The relationship between j and k is $k \leq 3j+1$. For performing the next prediction, starting in microrotation k, it is necessary to know the value of z in non-redundant representation. Starting with $j = 0$ we obtain $k = 1$. The second prediction is from $j = 1$ to $k = 4$. The third one is from $j = 4$ to $k = 13$. Consequently, the microrotations with indices $1, 4, 13, \ldots, j, 3j+1, \ldots$ must be repeated. In addition, the same number of updates of the z coordinate must be carried out using carry-save adders and its non-redundant representation must be obtained [13].

The prediction algorithm presents two drawbacks. In the first prediction it is only possible to obtain σ_0 and σ_1. In order to perform the next prediction it is necessary to update the z coordinate and convert it from a carry-save to a non-redundant representation. For preventing the latency from being determined by this conversion, the first microrotation has to be $j = 1$ [13]. This, however, constrains the convergence range in circular coordinates to [-1,+1]. On the other hand, in the z coordinate, we perform several updates in carry-save representation. After each update the conversion from a carry-save to a non-redundant representation is necessary in order to be able to perform the next prediction. This makes it impossible for the prediction to be implemented in pipelined architectures with high throughput. The prediction based architectures that have been proposed are unfolded but not pipelined [2][13].

3 Modified Parallel Prediction Algorithm

In this section we present a new CORDIC algorithm for the rotation mode with carry-save arithmetic based on the prediction of the σ_i coefficients that can be implemented in a high speed and low latency pipelined architecture.

Before describing the new prediction algorithm we must point out that in order to maintain a constant scale factor it is necessary that $\sigma_i \in \{-1,+1\}$. However for $i \geq n/4$ we can use $\sigma_i \in \{-1,0,+1\}$ maintaining a constant scale factor [13]. To this end, when $\sigma_i = 0$ (the microrotation is not carried out) the following scaling is carried out:

$$\begin{aligned} x_{i+1} &= x_i + 2^{-2i-1}x_i \\ y_{i+1} &= y_i + 2^{-2i-1}y_i \\ z_{i+1} &= z_i \end{aligned} \qquad (4)$$

The prediction algorithm we present is based on the use of radix-2 microrotations with multiple shifts and $\sigma_i \in \{-1,+1\}$ for $0 \leq i \leq n/4$, radix-2 microrotations with $\sigma_i \in \{-1,0,+1\}$ for $n/4 \leq i \leq \lceil (n-1)/3 \rceil$, and radix-4 microrotations with $\sigma_i \in \{-2,-1,0,+1,+2\}$ for $i > \lceil (n-1)/3 \rceil$.

3.1 Radix-2 Microrotations with Multiple Shifts

With the new prediction algorithm we carry out the prediction of the $\sigma_i's$ from $i = 0$ to $i = n/4$. This way we can obtain the next prediction using $\sigma_i \in \{-1,0,+1\}$ (As $i \geq n/4$, K can be kept constant). This permits performing the prediction with parallel operations and without any conversion from carry-save to non-redundant arithmetic. In order to obtain the simultaneous prediction of $\sigma_0, \sigma_1, \ldots, \sigma_{n/4}$ the condition (3) with $j = 0$ and $k = n/4$ must be met. As $\alpha_i = \tan^{-1}(2^{-i})$ condition (3) is not verified and the prediction cannot be carried out this way. In order to solve this problem, we select microrotation angles with multiple shifts, that is

$$\alpha_i = \tan^{-1}(2^{-i} + s_{i,1}2^{-d_{i,1}} + s_{i,2}2^{-d_{i,2}} + \ldots) \qquad (5)$$

with $s_{i,j} \in \{-1,0,+1\}$ and $i < d_{i,j} \leq n$. The values of $s_{i,j}$ and $d_{i,j}$ are selected in order to reduce the difference between 2^{-i} and α_i, and to meet condition (3)

i	$n = 16$	$n = 24$	$n = 32$
0	(0,1)	(0,1,4)	(0,1,4)
1	(1)	(1,5,6)	(1,4,−6)
2	(2)	(2)	(2,8,10)
3	(3)	(3)	(3)
4	(4)	(4)	(4)
5	(5)	(5)	(5)
6	(6)	(6)	(6)
i		$n = 40$	$n = 53$
0		(0,1,4,−8)	(0,1,4,−8,−10)
1		(1,5,6)	(1,4,−6,−11,−14)
2		(2,7,−9,−11)	(2,8,10,11,−15)
3		(3,11,13)	(3,11,13,15)
4		(4)	(4,14,16)
5		(5)	(5)
6		(6)	(6)

$(a, b, \ldots, -c)$ means $\tan^{-1}(2^{-a} + 2^{-b} \ldots - 2^{-c})$

Table 1: Microrotation angles for parallel prediction

for $j = 0$ and $k = n/4$. For example for the microrotation $i = 1$, $|\tan^{-1}(2^{-1}) - 2^{-1}| = 0.0363523$. This difference can be reduced introducing multiple shifts, that is, $|\tan^{-1}(2^{-1} + 2^{-5} + 2^{-6}) - 2^{-1}| = 0.000441$. For seeking this microrotation angles we also impose the condition of minimizing the number of shifts (the number of additions) in order to obtain the solution with the minimum hardware cost.

In table 1 we display the microrotation angles (only for $i \leq 6$) obtained for 16, 24, 32, 40 and 53 bit precision. As the precision increases, the number of multiple shifts grows due to the fact that it is necessary to predict a larger number of σ_i's without updating the z coordinate. Furthermore, we see that the multiple shifts are only needed in the first microrotations, where the difference between α_i and 2^{-i} is larger.

Simultaneously to the radix-2 microrotations with multiple shifts, we perform the updating of z in carry-save representation so that $z_{n/4+1}$ is obtained as a sum word ($z^s_{n/4+1}$) and a carry word ($z^c_{n/4+1}$). In order to obtain the next prediction, a redundant radix-2 signed-digit addition [5][12] of $z^s_{n/4+1}$ and $z^c_{n/4+1}$ is carried out. This way, we obtain $z_{n/4+1}$ in radix-2 signed-digit redundant representation with digits in the range $\{-1, 0, +1\}$. As in these microrotations $i \geq n/4$, we can use $\sigma_i \in \{-1, 0, +1\}$ while maintaining a constant scale factor. Then, the prediction of the σ_i's is obtained by simply assigning the digits of $z_{n/4+1}$ to the σ_i's. We avoid this way the conversions from carry-save to non-redundant arithmetic. For subsequent predictions we can use the same technique. However, in order to reduce the number of stages, we use radix 4 microrotations in the next prediction.

3.2 Radix-4 Microrotations

In order to reduce the number of microrotations of the CORDIC algorithm it is possible to combine two radix-2 microrotations to obtain one radix-4 microrotation when the approximation $\tan^{-1}(2^{-i}) = 2^{-i}$ is verified within n bit precision [3][10][13]. Then the σ_i coefficients are obtained by recoding in a similar way as the Booth multiplication, and take values from the radix-4 redundant digit set $\{-2, \ldots, 2\}$.

In [10] radix-4 microrotations are employed when $i > n/2$ in order to reduce the number of microrotations of the CORDIC algorithm while maintaining a constant scale factor. However, as for $i > (n-1)/3$ we have that $\tan^{-1}(2^{-i}) = 2^{-i}$ within a precision of n bits, in the CORDIC algorithm with parallel prediction we use radix-4 microrotations from $i = \lceil (n-1)/3 \rceil + 1$. Consequently, radix-2 microrotations are used for $0 \leq i \leq \lceil (n-1)/3 \rceil$ and radix-4 microrotations for $\lceil (n-1)/3 \rceil < i \leq \lceil (n-1)/3 \rceil + (n - \lceil (n-1)/3 \rceil)/2$. The shifts of the radix-4 microrotations are

$$q_i = 2i - (\lceil (n-1)/3 \rceil + 1) \qquad (6)$$

Observe that the shifts are radix-4, that is, the shift increases by two in each iteration. The number of microrotations of the CORDIC algorithm has been reduced from $n + 1$ to approximately $\lceil 2n/3 \rceil + 1$.

However, as the radix-4 microrotations start from $i = \lceil (n-1)/3 \rceil + 1$, the scale factor is not constant. The non-constant scale factor generated in the radix-4 microrotations from $q_i = \lceil (n-1)/3 \rceil + 1$ to $q_i = n/2$ must be incorporated to the constant and known beforehand factor that is generated in the radix-2 microrotations. In order to calculate the contribution of these radix-4 microrotations to the scale factor we use the approximation

$$(1 + \sigma_i^2 2^{-2q_i})^{-1/2} = 1 - \sigma_i^2 2^{-2q_i - 1} \qquad (7)$$

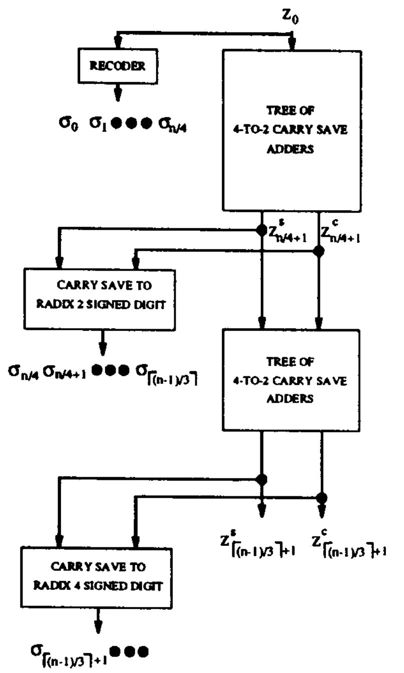
with $q_i > \lceil (n-1)/3 \rceil$, being q_i the radix-4 shift. This way, the final scale factor is obtained by multiplying the scale factor generated in the radix-2 microrotations times the factor $(1 - \sigma_i^2 2^{-2q_i - 1})$, which is the scale factor introduced in each radix-4 microrotation, by means of addition and shift operations controlled by the value of σ_i. This operation can be carried out in parallel with the microrotations.

It is interesting to note that in some cases is better to begin the radix-4 microrotations from $i = n/2 + 1$ to have a constant scale factor. For example for moderate n the reduction achieved by introducing radix-4 microrotations for $i \leq n/2$ seems to be low in comparison with the complexity of a non-constant scale factor. On the other hand to compute sines and cosines the final multiplication by the scale factor is avoided if this factor is constant, by just introducing $x_0 = 1/K$ and $y = 0$ [12]. Here we consider the more general case where the input coordinates are not known in advance and the scale factor must be compensated.

When we introduce radix-4 microrotations we make the approximation $\tan^{-1}(2^{-i}) = 2^{-i}$. This means that the circular CORDIC is approximated by the linear CORDIC in the rotation mode, so we are performing a Booth multiplication. Hence an alternative to the radix–4 microrotations (iterative multiplication) is to make a fast parallel multiplication (fast termination algorithm [1]). Here we consider the radix–4 stages but our algorithm also can incorporate the fast termination methods.

3.3 Parallel Prediction Algorithm

Figure 1 summarizes the parallel prediction algorithm we have developed. Initially, starting from the value of z_0, we carry out the prediction of $\sigma_0, \sigma_1, ..., \sigma_{n/4}$, taking values from the set $\{-1, +1\}$. In order to insure the convergence of the algorithm it is necessary to introduce multiple shifts in the evaluation of the x and y coordinates. Simultaneously we update the z coordinate in order to obtain $z_{n/4+1}$ in a carry–save representation. It is necessary to repeat microrotation $n/4$ in order to correct prediction errors.



z0
RECODER
σ0 σ1 ●●● σn/4
TREE OF 4-TO-2 CARRY SAVE ADDERS
z^s n/4+1
z^c n/4+1
CARRY SAVE TO RADIX 2 SIGNED DIGIT
σn/4 σn/4+1 ●●● σ⌈(n-1)/3⌉
TREE OF 4-TO-2 CARRY SAVE ADDERS
z^s ⌈(n-1)/3⌉+1
z^c ⌈(n-1)/3⌉+1
CARRY SAVE TO RADIX 4 SIGNED DIGIT
σ⌈(n-1)/3⌉+1 ●●●

Figure 1: New parallel prediction algorithm

To predict the coefficients $\sigma_{n/4}$, $\sigma_{n/4+1}$, $\ldots$, $\sigma_{\lceil(n-1)/3\rceil}$, we perform a signed digit radix–2 addition of the sum , $z^s_{n/4+1}$, and carry, $z^c_{n/4+1}$, words of $z_{n/4+1}$. This way, $z_{n/4+1}$ is coded into a signed digit radix–2 representation with digits in the set $\{-1, 0, 1\}$. From this value of the z coordinate we directly obtain the $\sigma_i's$ with $n/4 \leq i \leq \lceil(n-1)/3\rceil$, which can take values from the set $\{-1, 0, 1\}$. It is not necessary to introduce a correcting microrotation as the subsequent radix–4 microrotations correct the error in the prediction.

Later, z is evaluated again in order to obtain $z_{\lceil(n-1)/3\rceil+1}$ in a carry-save representation. $z_{\lceil(n-1)/3\rceil+1}$ is coded into a signed-digit radix-4 representation using the set of digits $\{-2, \ldots, +2\}$. A recoder of this type is described in [7]. This way, as the $\sigma_i's$ with $i \geq \lceil(n-1)/3\rceil + 1$ can take values from the set $\{-2, \ldots, +2\}$, the prediction is obtained setting the $\sigma_i's$ equal to the radix–4 digits of $z_{\lceil(n-1)/3\rceil+1}$. With the last prediction we obtain the $\sigma_i's$ up to the last microrotation (it is not necessary to reevaluate z).

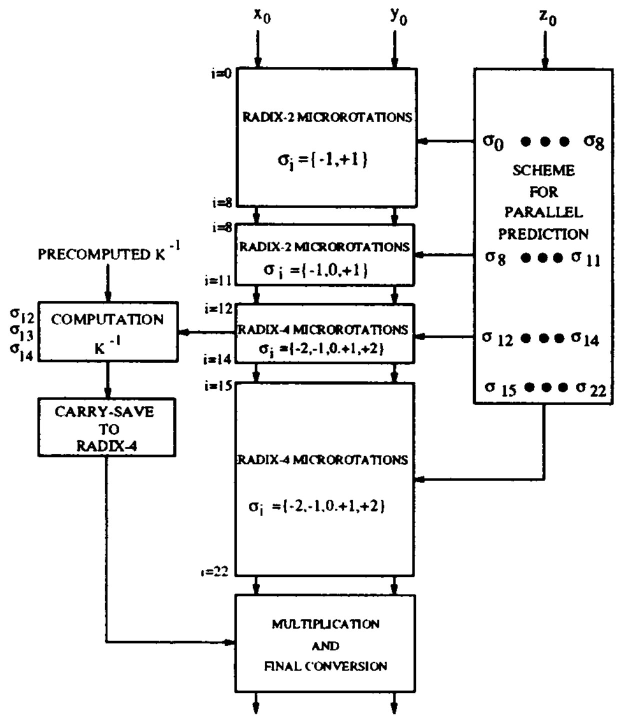


Figure 2: Architecture of the CORDIC processor

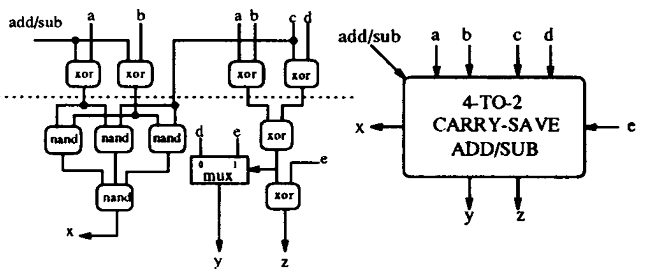


Figure 3: Slice of a 4-to-2 CSA

With the prediction algorithm we have developed, it is only necessary to make three predictions of the $\sigma_i's$, for $0 \leq i \leq n/4$, for $n/4+1 \leq i \leq \lceil(n-1)/3\rceil$ and for $i \geq \lceil(n-1)/3\rceil + 1$. In order to obtain these predictions it is not necessary to obtain the z coordinate in non-redundant representation. We have to recode the carry-save representation of z in a signed-digit radix-2 or radix-4 representation. In these codings there is no carry propagation, and they are thus fast and their hardware complexity is similar to that of a

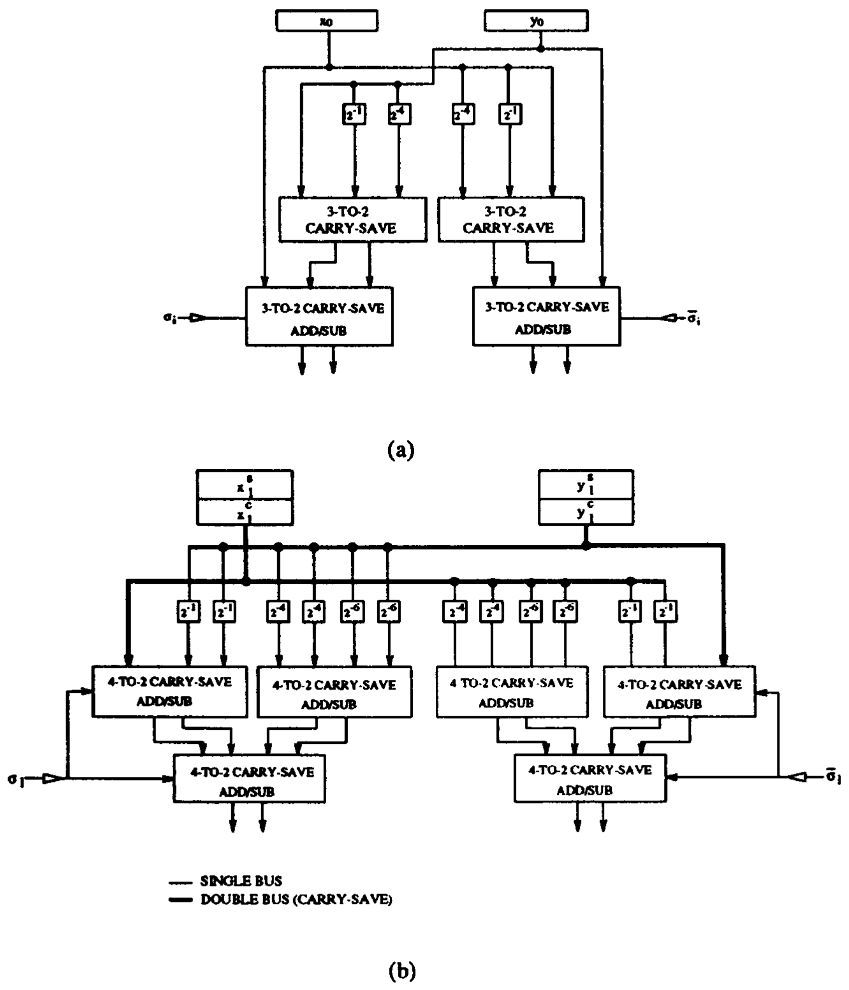


Figure 4: Radix–2 microrotation with multiple shifts

4–to–2 carry–save adder. In addition, it is only necessary to update coordinate z twice. These updates are carried out with tree structures of 4–to–2 carry–save adders. As all the operations are performed without carry propagations, it is possible to pipeline this architecture in order to obtain a high throughput.

4 Architecture

In figure 2 we display the architecture of the radix–2–4 CORDIC processor based on the parallel prediction for 32 bit precision. The microrotations from $i = 0$ to $i = 8$ are radix–2 with $\sigma_i \in \{-1, +1\}$ and multiple shifts (see table 1). Microrotation $i = 8$ must be repeated in order to correct the errors in the prediction of the σ_i coefficients. From $i = 8$ (repetition) to $i = 11$ we perform radix–2 microrotations with $\sigma_i \in \{-1, 0, +1\}$ and simple shifts. It is not necessary to repeat microrotation $i = 11$ because the prediction errors are corrected with the radix–4 microrotations.

In the radix–4 microrotations we can distinguish between two groups: microrotations 12, 13 and 14 for which the contribution to the final scale factor must be calculated and the remaining microrotations, 15, ..., 22, which have no bearing on the value of the scale factor. In parallel with microrotations 12, 13 and 14 there is a calculation of the scale factor in carry–save representation by means of three addition stages with hardwired shifts and multiplexers in order to choose the appropriate term as a function of the value of σ_i (see equation 7). The complexity of these stages is similar to the implementation of a coordinate in the radix–4 microrotation. In order to compensate the scale factor we convert the scale factor obtained to radix–4 signed–digit format. We subsequently carry out the multiplication in each of the coordinates, x and y, with two multipliers, one for each coordinate, organized in 4–to–2 carry–save adder trees. Finally there is a stage for the conversion from redundant to non–redundant arithmetic.

The implementation of the microrotations and the prediction algorithm is based on the use of 4–to–2 carry–save adder/subtracters. We have adapted the 4–to–2 carry–save adder described in [7] for the evaluation of 2's complement subtraction where two of the four operands must be complemented. The structure of a bit slice of the 4–to–2 carry–save adder/subtracter is shown in figure 3. Two XOR gates have been included in order to complement, as a function of the *add/sub* signal, bits a and b in the production of the transfer bit x. The production of the addition bit, z, does not change, as two of the four inputs are complemented, and the calculation of the carry bit, y, does not change either because we are using bit d and the transfer bit e. To complete the 2's complement we introduce a 1 in the e bit of the least significant bit slice and another 1 in the least significant bit of the carry output. The delay is still the same as in the case of a 4–to–2 carry–save adder.

In figure 4 we present the structure of the radix–2 microrotations with multiple shift and $\sigma_i \in \{-1,+1\}$. Figure 4(a) corresponds to the architecture of the first microrotation (for the case $n = 32$). For each coordinate we just have to add four operands (three hardwired shifts) because the input data are in non-redundant arithmetic. As three of the operands may be subtracted, we have had to use a 3–to–2 carry-save adder and a 3–to–2 carry–save adder/subtracter per coordinate. Figure 4(b) corresponds to a microrotation with three hardwired shifts in which the input operands are in redundant arithmetic. Although the hardware cost is three times that of the case of microrotations with a single shift, the delay is only double (two 4–to–2 carry–save adder/subtracter levels). On the other hand, the radix–2 microrotations with $\sigma_i \in \{-1,+1\}$ and simple shift are similar to the one shown in figure 4(b) with the difference that it uses a single 4–to–2 carry–save adder per coordinate as there is a single shift.

In figure 5 we display the architecture of a radix–2 microrotation with simple shift and $\sigma_i \in \{-1,0,+1\}$. The 2–to–1 multiplexers permit selecting the appropriate operands in order to alternatively carry out the microrotation when $\sigma_i = +1$ or $\sigma_i = -1$ or the scaling when $\sigma_i = 0$ so as to maintain a constant scale factor (see equation (4)).

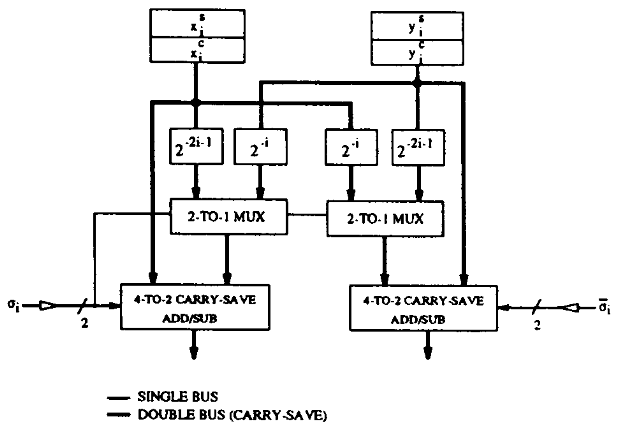


Figure 5: Redundant radix–2 microrotation

The structure of a radix–4 microrotation is similar to that of figure 5. In this case the 2–to–1 multiplexers are only used for the selection of the appropriate value of the shifted coordinate according to the value of σ_i (i.e. $0, 2^{-q_i}, 2 \cdot 2^{-q_i}$). We have assumed that the control inputs to the multiplexers are already decoded and that they are implemented with NAND gates [7]. This way there must be as many control inputs as operands to be selected. If all the control inputs are zero then no operands are selected and the output of the multiplexer is zero. When the control inputs attack a large number of multiplexers we have assumed the use of buffers in order to strengthen the signals.

This particular architecture for n=32 bits can be generalized to any precision. We must take into account that the number of stages with multiple shifts varies with the precision and so does the number of multiple shifts per microrotation (see table 1). However, the number of reevaluations of the z coordinate will be the same, that is, only two.

We have considered carry–save arithmetic in our design. However there are one conversion from carry-save to signed-digit in z coordinate. An alternative to carry–save arithmetic is to use radix–2 signed–digit [5][12], so this conversion will be avoided. As the complexity of this conversion is low we consider that in both cases (carry–save and signed–digit) the hardware cost should be similar.

4.1 Timing Analysis

The architecture based on the prediction algorithm we have presented can be pipelined in order to obtain a high throughput. In this architecture we obtain several $\sigma_i's$ in parallel in each prediction and the updating of the z coordinate is carried out with 4–to–2 carry–save adder trees. For this reason each σ_i is available before the corresponding microrotation has to be carried out. The delay will be determined by path x/y and not path z.

For the calculation of the delays we have used the data corresponding to the ES2 standard cell library for $1\mu m$ CMOS technology [8]. The delays are given based on the delay of a two input NAND gate (t_{nand-2}) and with a fanout of three NAND gates. A more detailed description of the delays assumed for each basic element can be found in [7]. We employ the following nomenclature: t_{reg} is the delay of a register, t_{2-1mux} is the delay of a 2–to–1 multiplexer, t_{4-2csa} is the delay of a 4–to–2 carry–save adder/subtracter and t_{buf} is the delay of a buffer.

We are going to consider two different pipelining levels of the architecture in order to obtain the delay of the slowest stage and the latency for each pipeline. Actually the processor can be pipelined at any desired level. We first evaluate the architecture with a pipeline scheme in which the delay of the slowest stages is similar to the delay of a radix–2 microrotation with simple shift (simple pipelining). After that we evaluate the architecture considering that each radix–2 microrotation with simple shifts is divided into two stages (double pipelining).

We denote as T_1 the delay of a radix–2 microrotation with $\sigma_i \in \{-1,+1\}$ and simple shift. T_2 is the delay of a radix–2 microrotation with $\sigma_i \in \{-1,0,+1\}$ and T_3 is the delay of a radix–4 microrotation.

a) *Simple pipelining.* Assuming that the architecture is pipelined so that the slowest stage is a microrotation with simple shift, the delay of the critical path will be given by the largest of T_1, T_2 and T_3.

$$\begin{aligned} T_1 &= t_{reg} + t_{buf} + t_{4-2csa} = 15.8 \\ T_2 &= t_{reg} + t_{buf} + t_{2-1mux} + t_{4-2csa} = 17.2 \quad (8) \\ T_3 &= t_{reg} + t_{buf} + t_{2-1mux} + t_{4-2csa} = 17.2 \end{aligned}$$

So the throughput of the processor is $17.2\,t_{nand-2}$. Consequently, the microrotations with multiple shifts must be pipelined in order to achieve the delay of a microrotation with simple shift. For this reason we pipeline the microrotations with multiple shifts into several stages where the delay of each stage is $t_{reg} + t_{buffer} + t_{4-2csa}$.

The latency is given by the total number of stages. This number does not coincide with the number of microrotations as we have pipelined the microrotations with multiple shifts. For n=32 bits we have a total of 24 microrotations, but 3 of them present triple shift (two of them pipelined into two stages) resulting in a latency of 26 cycles for carrying out the microrotations.

b) *Double pipelining.* For applications which require high speeds, we can pipeline the architecture so that the delay in the critical path is reduced by pipelining each microrotation with simple shift into two stages. For this we must pipeline the 4-to-2 carry-save adder/subtracter as shown in figure 3 (dashed lines). The microrotations with multiplexers for the selection of operands (radix-2 with $\sigma_i \in \{-1, 0, +1\}$ and radix-4) are pipelined so that the first stage is made up of the multiplexers and the first part of the 4-to-2 CSA, whereas the other stage is made up of the second part of the 4-to-2 CSA. This way the delays of all the stages are balanced.

If we denote as $t_{4-2csa(a)}$ the delay of the first part of the 4-to-2 CSA and $t_{4-2csa(b)}$ the delay of the second part, the delay of the slowest stage is

$$T = max \begin{cases} t_{reg} + t_{buf} + t_{2-1mux} + t_{4-2csa(a)} \\ t_{reg} + t_{4-2csa(b)} \end{cases} \quad (9)$$

This delay is $13.4\,t_{nand-2}$. With the double pipelining of each microrotation the delay of each stage is reduced by 22% with respect to the simple pipelining. This moderate reduction of the delay is due to the fact that the delay of the registers is very large as compared to the processing elements. The latency in this case will be double that of the previous case as we divide each one of the stages into two. Consequently, for n=32 we have a latency of 52 cycles.

5 Evaluation and Conclusions

In this section we evaluate the new prediction algorithm with respect to the one proposed in [13]. We also evaluate the architecture we propose with respect to a high speed CORDIC architecture that was recently proposed. When using the algorithm proposed in [13], the prediction can only be obtained from $i = 1$ on, and thus the convergence range in circular coordinates is constrained to $[-1, +1]$. On the other hand, in order to obtain the prediction, several conversions from carry-save to non-redundant representation are required for the z coordinate. In an unfolded architecture, such as the one proposed in [13], path x/y is slower than path z. However, it is not possible to pipeline the architecture in order to obtain a high throughput.

With the prediction algorithm we have presented, the convergence range in circular coordinates is larger than $[-\pi/2, +\pi/2]$, because the microrotations are started from $i = 0$ and multiple shifts have been introduced. On the other hand, the architecture can be pipelined in order to obtain a high throughput. Radix-4 microrotations are used in both designs. However, in [13] they incorporate two adder tree structures, one for each coordinate in order to maintain a constant scale factor. In the architecture we propose, the scale factor is calculated, by means of addition and shift operations, from the scale factor generated in the radix-2 microrotations, eliminating the tree structures in the x/y paths.

The prediction algorithm proposed here permits a high level of pipelining of the resulting architecture, making this architecture adequate for applications requiring high speeds. The CORDIC architecture for the rotation mode proposed in [4] is based on the calculation of the absolute value in MSD first mode of operation. The sign of z is propagated from the MSB to the LSB. There is no sign estimation and thus no additional microrotations are needed for insuring convergence.

In table 2 we compare this design with the one we present in this work. We have specified the cycle time, the latency and the hardware complexity with precisions 16, 32 and 53 bits. The hardware complexity is given in terms of the number of microrotations over x/y, the number of additions in z and the additional hardware required for each algorithm. In the microrotations with multiple shifts, we have assumed the equivalent number of microrotations with simple shift for the calculation of the hardware complexity (except in the first microrotation in which the input operands are not redundant). The additional hardware of the architecture proposed in [4] corresponds to the registers needed for delaying coordinates x and y, and for performing the skew of the z coordinate. The number of register rows needed for delaying x and y are the same as the wordlength of z, that is $n + log_2(n) + 2$. In the new architecture, the additional hardware corresponds to the number of addition and shift stages needed for the calculation of the scale factor in the first radix-4 microrotations. We have calculated the latency assuming double pipelining. For the calculation of the throughput we have taken into account that the critical path of the processor proposed in [4] is the sum of the delays for loading a register, the delay of a full-adder and the delay of an element for the calculation of the absolute value.

The cycle time of the architecture we have presented is less than that of the architecture in [4]. Even if in our case we consider simple pipelining the cy-

Design	stage delay	precision	latency	Microrot. x/y	Additions in z	Additional rows of reg.	Additional shift+add.
Dawid & Meyr [4]	16.1	16	56	17	17	22(x/y) and 11 (z)	–
		32	105	33	33	39(x/y) and 20 (z)	–
		53	169	54	54	61(x/y) and 31 (z)	–
New design	13.4	16	26	13	7	–	2
		32	52	28	13	–	3
		53	90	52	20	–	5

Table 2: Comparison with a high speed CORDIC processor

cle time would be similar, but with a very important reduction in area due to the removed registers. On the other hand, the number of stages has been almost halved and, consequently, the latency is significantly smaller in our design. The hardware cost of our architecture is lower. This is due to the fact that the architecture proposed in [4] must include many register rows in order to delay the x and y coordinates and to skew the z coordinate. We must take into account that the area of a register cell is around 67 % the area of a 4–to–2 carry–save adder [7]. In addition, the prediction algorithm makes the number of additions in the z coordinate small. On the other hand, the algorithm proposed in [4] can only use radix–2 microrotations with $\sigma_i \in \{-1,+1\}$, and thus it is not possible to reduce the number of microrotations by including radix–4 microrotations.

Consequently, the algorithm we have presented for the prediction of the coefficients of the microrotations of the CORDIC algorithm, permits the design of an architecture for the rotation mode presenting a low latency and a high speed. The introduction of radix–4 and radix–2 microrotations with coefficients $\sigma_i \in \{-1,0,+1\}$ has led to a reduction of the latency and the hardware with respect to other implementations of the CORDIC algorithm. On the other hand, the introduction of multiple shifts in the first microrotations in order to facilitate the prediction has led to a highly pipelined architecture, achieving a cycle time that is lower than other high speed implementations.

Acknowledgments

The authors would like to thank the referees for their valuable comments.

References

[1] H.M. Ahmed, "Efficient elementary function generation with multipliers", Proc. 9th Symp. on Computer Arithmetic, pp. 52–59, 1989.

[2] P.W. Baker, "Suggestion for a fast binary sine/cosine generator", IEEE Transactions on Computers, pp. 1134-1136, 1976.

[3] J.D. Bruguera, E. Antelo and E.L. Zapata, "Design of a pipelined radix–4 CORDIC processor", J. Parallel Comput., vol.19, no 7, pp. 729-744, 1993.

[4] H. Dawid and H. Meyr, "High speed bit–level pipelined architectures for redundant CORDIC implementation", Proc. ASAP'92, pp. 358-72, 1992.

[5] J. Duprat and J.-M. Muller, "The CORDIC algorithm: New results for fast VLSI implementation", IEEE Trans. on Comput., vol 42, no 2, pp. 168-78, 1993.

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

[7] M. Ercegovac and T. Lang, "Division and Square Root: Digit–Recurrence, Algorithms and Implementations", Kluwer Academic Pub, 1994.

[8] European Silicon Structures,"ES2 ECPD10 Library Databook", June 1992.

[9] Y.H. Hu, "CORDIC–based VLSI architecture for Digital Signal Processing", IEEE Signal Processing Magazine, no 7, pp. 16-35, July 1992.

[10] J.-A. Lee and T. Lang, "Constant–Factor redundant CORDIC for angle calculation and rotation", IEEE Transactions on Computers, vol. 41, no 8, pp. 1016-1025, Aug. 1992.

[11] T.G. Noll, "Carry–save architectures for high speed signal processing", J. VLSI Signal Processing, vol. 3, pp. 121-140, 1991.

[12] N. Takagi, T. Asada and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation", IEEE Trans. on Comput., vol. 40, no 9, pp. 989-995, 1991.

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

[14] J.E. Volder, "The CORDIC trigonometric computing technique", IRE Trans. Elect. Comput., vol. EC-8, pp. 330-334, Sept. 1959.

[15] J.S. Walther, "A unified algorithm for elementary functions", Proc. Spring. Joint Comput. Conf., pp. 379-385, 1971.