On-line CORDIC algorithms

Haixiang Lin

IBBC-TNO Organization for Applied Scientific Research
Rijswijk, The Netherlands

Henk J. Sips

Delft University of Technology
Delft, The Netherlands

abstract

The CORDIC algorithms provide in a fast way the calculations of a number of arithmetic basic functions. A CORDIC calculation takes $O(n)$ steps for a function, where n is the word length of the operands. The speed is limited by the carry propagation in the adders and the I/O throughput. Speed can be improved by introducing redundancy in the calculation circuitry and I/O throughput by doing I/O transfers while calculating. The latter is characteristic for the class of so called on-line arithmetic. At the same time the pin requirements are limited to a single digit per operand. This paper introduces a number of new algorithms to make an on-line CORDIC implementation.

1. Introduction

In on-line computations, the input operands and results flow through arithmetic units in a digit by digit manner, starting with the most significant digit. An on-line algorithm is said to have an on-line delay of δ, if for the generation of the j-th digit of the result, $(j+\delta)$ digits of the input operands are required [1]. Fig.1 shows the principle of on-line computation and illustrates that digit on-line arithmetic units can be chained to obtain a fast throughput of digit-serial operands. In recent years many algorithms for the on-line generation of the basic arithmetic functions: addition, multiplication, division, and square root have been considered [2-8]. In essence, two algorithmic models have been used to find suitable algorithms for the on-line calculation of an arithmetic function. The first model is based on the use of recurrence equations. In this model a partial result (or an equivalent function) is calculated each iteration step by using the partial result from the previous cycle [2,3,4,5]. The second model is based on function minimization. The function to be calculated is transformed to an equivalent function which has to be minimized each iteration step. Most of the algorithms in this model are based on the use of continued sum/product (CSP) algorithms [9]. Examples can be found in [6,7,8].

The on-line generation of transcedental functions turns out to be more difficult. A traditional known fast method for calculating these functions is the CORDIC method [11,12]. The CORDIC algorithms are of the CSP-type and have $O(n)$ timing characteristics, where n is the word length of the operands. In this paper it is shown that the CORDIC algorithms can be modified such, that the operations are on-line with respect to input- and output operands.

2. The CORDIC equations.

The CORDIC method is based on vector rotations [11,12], where a new vector $P_{i+1}=(X_{i+1},Y_{i+1})$ is obtained from $P_i=(X_i,Y_i)$ according to

$$X_{i+1} = X_i + m\rho_i Y_i 2^{-i} \qquad (1a)$$

$$Y_{i+1} = Y_i - \rho_i X_i 2^{-i} \qquad (1b)$$

$$Z_{i+1} = Z_i - \rho_i \theta_i \qquad (1c)$$

where $m \in \{-1,0,1\}$ is the parameter for the coordinate system, and $\rho_i \in \{-1, 1\}$. Although the equations can be formulated for a general radix, we will restrict the discussion to $r=2$, for reasons of convenience. Various arithmetic functions can be calculated by forcing Z or Y to zero, by chosing the appropriate values of ρ_i in each iteration cycle. The values of θ_i are found by $\theta_i=m^{-1/2}\tan^{-1}[m^{1/2}\cdot 2^{-i}]$. The results calculated in Eq.(1a,1b) have to be corrected through multiplication by the following scaling factor

$$K^{-1} = \prod_{i=0}^{n-1}(1+m\phi_i^2)^{-1/2} \qquad \text{with} \quad \phi_i=\rho_i 2^{-i} \qquad (2)$$

For the traditional CORDIC algorithms this scaling factor is constant and can be calculated in advance. Some other schemes use a slight modification of the iteration scheme in order to simplify the factor (e.g., to force the factor to be a power of 2), such that a multiplication is not necessary and the corrections can be done by simple additions in each iteration [13, 14].

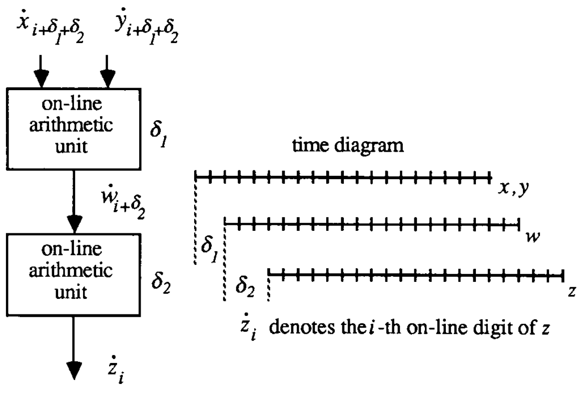


Fig.1 Chained on-line computation

A number of authors have published on-line versions of some of the CORDIC algorithms. Owens [8] uses a simplified CORDIC scheme to calculate $sin(x)$ with an on-line delay of $\delta=3$ for $r=8$. However, the result is unscaled (Eq.(2)), so it cannot be used directly for further calculations. Ercegovac and Lang [15] describe two chained CORDIC schemes to calculate rotation factors, e.g $sin(\theta)$, $cos(\theta)$, with $\theta=tan^{-1}(x/y)$. The first CORDIC scheme computes the angle which is transmitted in decomposed form, i.e. a sequence of ρ_i values, to the CORDIC module which computes the rotation. The total latency time of their implementation is $3n+3$ clock cycles.

3. The On-line CORDIC equations

For on-line processing, the operands are to be represented by a radix-r redundant digit set $(-\eta,..,-1,0,1,..,\eta)$ where η is the redundancy factor $(r/2\leq\eta\leq r-1)$. An operand X is recursively defined as

$$X_i = X_{i-1} + x_{i+\delta}\cdot r^{-i-\delta-1} \qquad (3)$$

and

$$X_0 = \sum_{j=0}^{\delta-1} x_j\cdot r^{-j-1}$$

where δ denotes the on-line delay. Since we restrict ourselves in this paper to $r=2$, the fully redundant digit set $\{-1,0,1\}$ is the obvious result to work with.

In the on-line case the CORDIC equations of (1) are transformed to the following set of equations

$$X'_{i+1} = X'_i + m\rho_i Y'_i\cdot 2^{-i}+\Delta X_{i+1}(x_{i+\delta}, y_{i+\delta}) \qquad (4a)$$

$$Y'_{i+1} = Y'_i - \rho_i X'_i\cdot 2^{-i}+\Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta}) \qquad (4b)$$

$$Z'_{i+1} = Z'_i - \rho_i\theta_i + z_{i+\delta}\cdot 2^{-(i+\delta)} \qquad (4c)$$

where $x_{i+\delta}$, $y_{i+\delta}$, and $z_{i+\delta}$ are the new input digits of X, Y, and Z, respectively, at the i-th iteration. X'_0, Y'_0, and Z'_0 are set to X_0, Y_0, and Z_0, respectively, according to Eq.(3). The terms ΔX_{i+1} and ΔY_{i+1} are the correction terms to correct the error due to the absence of $x_{i+\delta}$, $y_{i+\delta}$, and $z_{i+\delta}$ in the iterations $0,1,..,i-1$. The calculation of ΔX_{i+1} and ΔY_{i+1} is considered in Appendix A.

4. On-line characteristics for $Z\rightarrow 0$

To compute the arithmetic functions like $x\cdot cos(z)-y\cdot sin(z)$ $(m=1)$, or $x\cdot cosh(z)+y\cdot sinh(z)$ $(m=-1)$, Z'_i is forced to zero.

If we define $W_i=2^i\cdot Z'_i$, Eq.(4c) can be rewritten as

$$W_{i+1} = 2(W_i + z_{i+\delta}\cdot 2^{-\delta}- \rho_i\cdot\alpha_i) \qquad (5)$$

where $\alpha_i=2^i\cdot\theta_i$ can be directly read from the constant angles table. The value of ρ_i is determined by $W_i + z_{i+\delta}\cdot 2^{-\delta}$. Since full carry propagation in an addition will be avoided, only the most significant L digits of $W_i + z_{i+\delta}\cdot 2^{-\delta}$, denoted as W^*_i, are inspected. It holds that

$$|W_i + z_{i+\delta}\cdot 2^{-\delta} - W^*_i| < 2^{-(L-\varphi-1)} \qquad (6)$$

where φ is the position of the most significant digit of W_i. Combining Eqs. (5) and (6) it follows that

$$|W_{i+1}| < 2\cdot|W^*_i - \rho_i\cdot\alpha_i| + 2^{-(L-\varphi-2)} \qquad (7)$$

We must find an appropriate selection of ρ_i such that Z'_{i+1} converges. A selection function of the form

$$\rho_i = \begin{cases} 1 & \text{if } W^*_i \geq b \\ 0 & \text{if } |W^*_i| < b \\ -1 & \text{if } W^*_i \leq -b \end{cases} \qquad (8)$$

where b is a positive constant $(0<b\leq 1)$. For the value of b a simple value, like some power of two, is to be preferred to simplify the selection process. The choice of b must be such that Z'_{i+1} converges to zero, i.e. W_{i+1} is bounded by a fixed positive constant.

First, we consider the case of circular rotation, i.e. $m=1$. It can be shown that to ensure the convergence of Z'_{i+1} it requires $\delta=2$. Furthermore, with $\delta=2$, $b=1/2$ and $L=5$ $(\varphi=0)$, it holds $|W_i| < 2$ for all i. The proof of this is lengthy, but straightforward, so it suffices to give the procedure. W_0 consists of the first two digits of the input Z, therefore it holds $|W_0|\leq 1.5$. By considering all possible combinations of the first four most significant input digits, it follows that $|W_1|\leq 1.93$ an $|W_2|\leq 1.51$. Now $\theta_i=tan^{-1}[2^{-i}]$, so by using a Taylor series-expansion which is truncated and rounded up- and downwardly, it holds that

$$2^{-i}-(1/3)2^{-3i}+(1/35)2^{-5i}\leq \theta_i \leq 2^{-i}-(1/3)2^{-3i}+(1/5)2^{-5i}$$

or

$$1-(1/3)2^{-2i}+(1/35)2^{-4i}\leq \alpha_i \leq 1-(1/3)2^{-2i}+(1/5)2^{-4i}$$

By using this, it can be shown by induction that

$$2\cdot|W^*_i - \rho_i\cdot\alpha_i| + 2^{-(L-2)}\leq 1.51, \text{ if } |W_i| < 1.51 \text{ for } i\geq 2.$$

From the definition $W_i=2^i\cdot Z'_i$ and $|W_i| < 2$, it follows that $Z'_{n+1}<2^{-(n-1)}$. Thus, the convergence of the angle rotation of Eq.(4c) is guaranteed for $m=1$.

For $m=-1$ (hyperbolic rotation), a similar analysis as for $m=1$ can be made. It can be shown that using the same selection function (Eq.(8)) it holds $|W_i| < 1.5$ $(|W_0|<1.0)$, for $\delta=2$ and $L=4$.

5. On-line characteristics for $Y\rightarrow 0$

To compute the functions $z+tan^{-1}(y/x)$ and $(x^2+y^2)^{1/2}$ (when $m=1$), or $z+tanh^{-1}(y/x)$ and $(x^2-y^2)^{1/2}$ (when $m=-1$), Y has to be forced to zero. For the Eq.(4b), it holds that the convergence is guaranteed if $|Y_{i+1}| \leq |X_i|\cdot 2^{-i}$ (X_i is bounded, see [12]). The equation to be forced to zero is Eq.(4b), i.e.

$$Y'_{i+1} = Y'_i - \rho_i\cdot X'_i\cdot 2^{-i}+\Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta}) \qquad (9)$$

The choice of ρ_i is determined by the value of Y'_i. Let $V_i=2^i\cdot Y'_i$, and assume again that only the L most significant digits of V_i are used for the selection and denote them as V^*_i. Then it holds that $V_i = V^*_i+\varepsilon(V)$, with $\varepsilon(V)$being the truncation error and $|\varepsilon(V)|\leq 2^{-(L-1)}$.

The selection function for ρ_i is defined as

$$\rho_i = \begin{cases} 1 & \text{if } V^*_i \geq b \\ 0 & \text{if } |V^*_i| < b \\ -1 & \text{if } V^*_i \leq -b \end{cases} \qquad (10)$$

where b is a positive constant ($0<b\leq 1$). The selection of can simply be done by comparing the L most significant digits of V^*_i with b. The value of b must be chosen such that the convergence criterion, $|Y_{i+1}| \leq |X_i| \cdot 2^{-i}$, is satisfied. For the circular CORDIC, i.e. $m=1$, two cases can be considered:

A. $\rho_i=0$

This implies that $|V^*_i|<b$ or $|Y'_i|<b\cdot 2^{-i}+2^{-(i+L-1)}$. Since (see Appendix B) it holds

$$\left| \sum_{k=i+\delta}^{n} \Delta Y_{i+1}(x_k,y_k) \right| < 2.65 \cdot 2^{-(i+\delta+1)} \qquad (11)$$

therefore,

$$|Y_{i+1}| = |Y'_i + \sum_{k=i+\delta}^{n} \Delta Y_{i+1}(x_k,y_k)| \leq b \cdot 2^{-i} + 2^{-(i+L-1)} + 2.65 \cdot 2^{-(i+\delta+1)} \qquad (12)$$

Since $X_i \geq 1/2$ [15] for $i \geq 1$, from Eq.(12), the condition $|Y_{i+1}| \leq X_i \cdot 2^{-i}$ is satisfied if

$$b \cdot 2^{-i} + 2^{-(i+L-1)} + 2.65 \cdot 2^{-(i+\delta+1)} \leq (1/2) \cdot 2^{-i}$$

Therefore, the values of the parameters b, δ, and L must be chosen to satisfy the following inequality

$$b \leq 1/2 - 2^{-(L-1)} - 2.65 \cdot 2^{-(\delta+1)} \qquad (13)$$

B. $\rho_i=\pm 1$

Assume that $\rho_i=+1$, i.e., $V^* \geq b$ (the same result follows for $V^* \leq -b$). With the substitution of

$$X'_i = X_i - \sum_{k=i+\delta}^{n} \Delta X_i(x_k,y_k)$$

into Eq.(4b), it follows

$$Y_{i+1} = Y'_i - X_i \cdot 2^{-i} + \sum_{k=i+\delta}^{n} [\Delta X_i(x_k,y_k) \cdot 2^{-i} + \Delta Y_{i+1}(x_k,y_k)] \qquad (14)$$

It can been shown that

$$\left| \sum_{k=i+\delta}^{n} [\Delta X_i(x_k,y_k) \cdot 2^{-i} + \Delta Y_{i+1}(x_k,y_k)] \right| \leq 3 \cdot 2^{-(i+\delta)} \quad \text{for } i \geq 0$$

(see Appendix B). Thus, from Eq.(14) it holds

$$Y_{i+1} \geq b \cdot 2^{-i} - 2^{-(i+L-1)} - X_i \cdot 2^{-i} - 3 \cdot 2^{-(i+\delta)}$$

and

$$Y_{i+1} \leq b \cdot 2^{-i} + 2^{-(i+L-1)} - X_i \cdot 2^{-i} + 3 \cdot 2^{-(i+\delta)}$$

Therefore, it holds $|Y_{i+1}| \leq X_i \cdot 2^{-i}$, if

$$b \geq 2^{-(L-1)} + 3 \cdot 2^{-\delta} \qquad (15)$$

and

$$b \leq 1 - 2^{-(L-1)} - 3 \cdot 2^{-\delta} \qquad (16)$$

From the analysis in A and B, it can be concluded that the choice of $b=1/4$, $\delta=4$, and $L=5$ satisfies the convergence criterion.

For $m=-1$, a similar analysis can be made. For arbitrary values of the inputs no fixed on-line delay can be derived, due to the characteristics of the resulting arithmetic function. However for special cases a better result can be given. As an example, for the function $ln(w)$, it holds $ln(w)=2 \cdot tanh(Y/X)$ with $X=w+1$ and $Y=w-1$, restricting the ranges of the operands to $1.5 \leq X < 2.0$ and $-0.5 \leq Y < 0.0$ for $0.5 \leq w < 1.0$. It can be shown that with $b=3/4$, $\delta=3$, and $L=5$ the convergence criterion is satisfied.

6. Digitizing the iteration results X'_i, Y'_i, and Z'_i

The values of X'_i, Y'_i and Z'_i are the results of the CORDIC calculation. These values are stored in redundant form and are driven to their final value in an iterative way and do not directly give the $(i-\delta)$-th digit at the i-th iteration. In order to produce an on-line result, these values must be transformed into an incremental representation where each iteration a new digit of the result is produced. This procedure is called digitization (e.g. see [7]).

A. The digitization of X'_i and Y'_i

Let $\dot{Y}_{i-d}$ (the same consideration holds for $\dot{X}_{i-d}$) be the on-line digitized result of Y'_{i+1} and $R_{i+1} = 2^i \cdot (Y'_{i+1} - \dot{Y}_{i-d})$ be the (scaled) residue (d is the on-line delay for digitization). Suppose $\dot{y}_0$ has a positional weight of $2^{\varphi+d}$ after d initial shifts (not to be confused with the real weight 2^{φ}), then by using a symmetrical rounding function up to the $(\varphi+d)$-th digit we obtain

$$\dot{y}_{i-d} = \text{Round}_{\varphi+d}(R_i) \qquad i \geq d,$$
$$\dot{y}_{i-d} = 0 \qquad 0 \leq i < d$$

$$R_{i+1} = 2 \cdot R_i + D(Y'_i) - 2^{\varphi+d} \cdot \dot{y}_{i-d} \qquad (17)$$

where

$$D(Y'_i) = 2^i \cdot (Y'_{i+1} - Y'_i) = -\rho_i \cdot X'_i + 2^i \cdot \Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta}).$$

It can be ensured that

$$|2R_i - 2^{\varphi+d} \cdot \dot{y}_{i-d}| < (1/2) 2^{d+\varphi} + 2^{d+\varphi-L+1}, \quad (L \geq 2) \qquad (18)$$

What remains is that the digitizer delay d and the truncation length L must be chosen such that $\dot{y}_{i-d}$ is a single digit.

For $m=1$, it holds that $|X'_i| \leq sqrt(2) \cdot K \leq 2.34$, for $|X|<1$ and $|Y|<1$, and $|\Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta})| \leq 2.65 \cdot 2^{-(i+\delta+1)}$ (see Appendix B). With $\delta=2$, it follows $|D(Y'_i)| \leq 2.34 + 2.65 \cdot 2^{-3} < 2.67$.

Therefore, it holds with $\varphi=1$ (remember that φ denotes the position of the most significant digit) that

$$|2R_i+D(Y'_i)-2^{\varphi+d}\cdot\dot{y}_{i-d}|<(1/2)\cdot 2^{d+1}+2^{d-L+2}+2.67 \qquad (19)$$

$\dot{y}_{i+1-d}$ is a single digit, if it holds that

$$|R_{i+1}|=|2R_i+D(Y'_i)-2^{\varphi+d}\cdot\dot{y}_{i-d}|<1.5\cdot 2^{d}-2^{d-L+1}$$

Therefore, the on-line condition becomes

$$(2.67\cdot 2^{-d}+3\cdot 2^{-(L-1)})\leq 1/2.$$

This inequality is satisfied by chosing $d=3$ and $L=5$.

For $m=-1$, it can be shown in a similar way that the conditions are satisfied by chosing $d=2$ and $L=6$.

B. The digitization of Z'_i

In case of $Y'_i \to 0$, the result Z'_i needs to be digitized for on-line output.
Let $R_i=2^i\cdot(Z'_i-\dot{Z}_{i-1-d})$, we use a slight different value for determining $\dot{z}_{i-d}$, which is,

$$\dot{z}_{i-d}=\text{Round}_{\varphi+d}(R_i+z_{i+\delta}\cdot 2^{-\delta}) \qquad i\geq d$$
$$\dot{z}_{i-d}=0 \qquad 0\leq i<d$$

$$R_{i+1}=2(R_i+z_{i+\delta}\cdot 2^{-\delta})-\rho_i\cdot\alpha_i-2^{\varphi+d}\cdot\dot{z}_{i-d} \qquad (20)$$

For $m=1$, it holds $|\alpha_i|=2^i\cdot|\tan^{-1}(2^{-i})|\leq 1$ for $i\geq 0$. With $\varphi=1$ ($|Z'_n|\leq\pi/2$), it follows,

$$|R_{i+1}|\leq|2(R_i+z_{i+\delta}\cdot 2^{-\delta})-2^{\varphi+d}\cdot\dot{z}_{i-d}|+|\alpha_i|$$
$$\leq(1/2)\cdot 2^{d+1}+2^{d-L+2}+1 \qquad (21)$$

$\dot{z}_{i+1-d}$ is a single digit, if

$$|2(R_i+z_{i+\delta}\cdot 2^{-(i+\delta)})|\leq 1.5\cdot 2^{d+1}-2^{d-L+2}$$

From Eq.(21) the on-line condition becomes

$$3\cdot 2^{-L+2}+2^{-d+1}+2^{-d-4}\leq 1$$

With $d=2$ and $L=5$ the on-line condition is satisfied.

For $m=-1$, it holds $|\alpha_i|=2^i\cdot|\tanh^{-1}(2^{-i})|\leq(10/9)$ for $i\geq 1$. With $\varphi=0$ ($|Z'_n|\leq\ln(1)$) and $Z=0$, the on-line condition is satisfied if $3\cdot 2^{-(L-2)}+(10/9)\cdot 2^{-(d-1)}\leq 1$. With $d=2$ and $L=5$ this condition is satisfied.

7. On-line generation of the scaling factor K^{-1}

To obtain the final on-line output, the digitized results $\dot{X}_i$ and $\dot{Y}_i$ must be multiplied with the scaling factor K^{-1}. Unlike in conventional CORDIC, K^{-1} is not a constant since ρ_i can be zero in the on-line case. This implies that the scaling factor cannot be calculated in advance. A method for calculating K^{-1} has been described Ercegovac and Lang [15]. They compute K^2 by the recurrence equation $K^2_{j+1}=K^2_j+m\rho_i^2\cdot K^2_j$. In the implementation the recurrence is unfolded by using $n/2$ stages of on-line adders. These stages are followed by an on-line square-rooter and -divider to obtain K^{-1}. Due to the unfolding, a relatively large on-line delay results, which in their implementation does not give an extra delay because it fits with the timing characteristics of the rest of the calculations.

Here a different method of calculating K^{-1} is shown. The method uses the table look-up approach, as described in [16]. This method has already been used in starting of an iterative method for calculating the reciprocal function [17]. The method is based on feeding the successive digits of the input operands to a table, while the table output produces an on-line digit each iteration cycle. If the number of input digits is large, soon a very large table would be required. A method to reduce the table requirements is to divide the product terms of the scaling factor in groups (partial scaling factors) and to generate each partial scaling factor with the aid of a table look-up system. The partial scaling factors can then be combined to give the final function. Fig. 2 shows this principle of operation for K^{-1}.

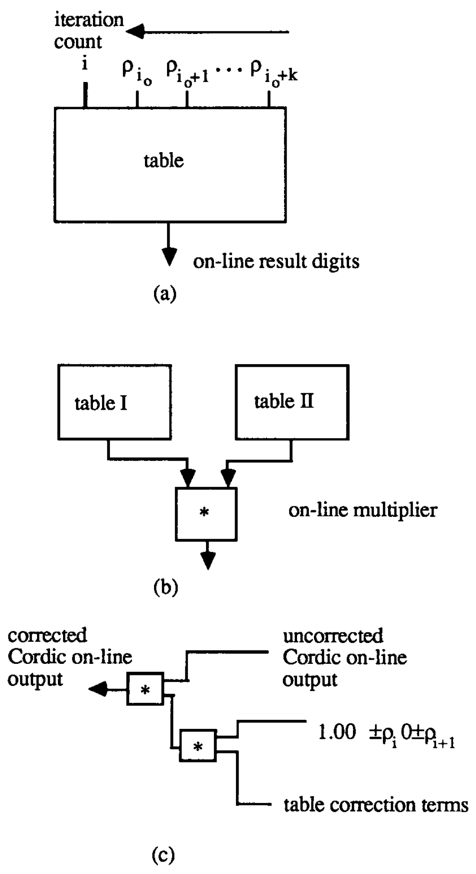

iteration count
i
ρ_{i_0} ρ_{i_0+1} ... ρ_{i_0+k}
table
on-line result digits
(a)
table I
table II
*
on-line multiplier
(b)
corrected Cordic on-line output
uncorrected Cordic on-line output
*
*
1.00 $\pm\rho_i$ $0\pm\rho_{i+1}$
table correction terms
(c)

Fig.2 a. Table generation of (partial) correction factor, b. Generation of the correction factor from the partial tables, c. Correction factor hardware organization.

It can be shown that by using theorem 1 in [16] the (partial) function

$$K^{-1}(i_0,p) = \prod_{i=i_0}^{p}(1+m\phi_i^2)^{-1/2} \tag{22}$$

can be generated with an on-line delay of $\delta=1-2i_0$, if $m=1$ and $\delta=2-2i_0$, if $m=-1$. Since $i_0=0$ for $m=1$ and $i_0=1$ for $m=-1$, it follows that K^{-1}can be generated with an on-line delay of $\delta=0$.

Now consider the case of splitting K^{-1} into 2 subproducts,

$$K^{-1} = \prod_{i=i_m}^{i_0-1}(1+m.\phi_i^2)^{-1/2}.\prod_{i_0}^{n-1}(1+m.\phi_i^2)^{-1/2} \tag{23}$$

where $i_m=0$, if $m=1$ and $i_m=1$, otherwise

From the results above it follows that a negative on-line delay results when generating the second partial scaling factor in Eq.(23) for $i_0 \geq 2$. This implies that the first $2i_0-1$ (or $2i_0-2$ for $m=-1$) digits of the second partial scaling factor are known before the first ρ_i becomes known. It can be shown that we have a number of the form $0.11...11xx$ or $1.0....0\bar{1}xx$ ($\bar{1}$ means -1) with $2i_0-1$ known digits for $m=1$. The latter representation might be advantageous in a multiplier. For $m=-1$ the subproduct has the form $1.00...00xx$ with $2i_0-2$ known digits.

The splitting according to Eq.(23) can be further performed in the same way. The final scaling factor is formed by multiplying the subproducts. A single partial scaling factor can be easily generated on-line by using a table with the values of $\rho_{i0}\ \rho_{i0+1}\ ..\rho_{i0+k}$ as address, plus an iteration counter i (see Fig.2a). Subproducts are combined by using on-line multipliers (one for two subproducts, two for three subproducts etc., see Fig.2b).

For an accuracy of 2^{-L}, only $L/2$ product terms of Eq.(2) are required. A further simplification can be obtained by observing that the higher ($L/4$) half of the $L/2$ product terms ($i=L/4+1,..., L/2$) can be reduced in complexity by deleting all cross products, because they are beyond the required accuracy, i.e. these $L/4$ product terms can be replaced by a single word of the form $1.00...(\pm\rho_i)0\,(\pm\rho_{i+1})0..$ The memory requirements are then determined by the partition sizes of the lower $L/4$ product terms ($i=i_m,..., L/4$). The size of the memory is $2i\cdot 2^p$, where p is the number of bits in the address space, i is the iteration counter. There are two output digits of the table. However, the signs are known in advance, so only two output bits suffice.

Table I shows the relation between the required memory and the number of (partial factor) partitions as a function of the required precision and Fig. 2c shows the hardware organization. The memory requirements can be traded off with multipliers. To rescale one CORDIC output (X_i) 1, 2, or 3 on-line multipliers are required for 1, 2, or 3 table partitions. For the second CORDIC output (Y_i) another on-line multiplier is needed. Since the generation of the partial scaling factors can all start at the first iteration, the resulting on-line delay of the table look-up phase is $\delta=0$. The total on-line delay of scaling factor is equal to the on-line delay of the on-line multiplier(s) for forming the partial scaling factors into one. The on-line delay of an on-line multiplier is 1 ([1], [16]). Therefore, for partition of 2 or 3 partial tables, the on-line delay for forming the scaling factor is only 1 to 2. Since $\delta_{input} + \delta_{digitizer} > 4$, the only additional on-line delay to the on-line CORDIC algorithm is the on-line delay of the multiplication of the scaling factor with the digitized result (see Fig.3), i.e. $\delta_{scaling}=1$.

8. Results and Conclusion

From the previous sections it follows that all applicable CORDIC operations can be made on-line. The total latency per function is given by

$$T_{tot} = n + \delta_{input} + \delta_{digitizer} + \delta_{scaling}.$$

The results $\delta_{total} = \delta_{input} + \delta_{digitizer} + \delta_{scaling}$ have been summarized in table II. In this table the columns denote the on-line inputs. So XYZ in column 1 denotes that X, Y, and Z are input on-line, while XY in column 2 denotes that only X and Y are input on-line. (Notice that $\delta_{scaling}=0$ for $\dot{Z}$ since no rescaling is required).

function	on-line delays X,Y,Z	X,Y	Z
$x.\sin\phi+y.\cos\phi$	6	4	6
$x.\sinh\phi+y.\cosh\phi$	5	3	5
$\tan^{-1}(y/x)$	6	6	2
$(x^2+y^2)^{1/2}$	8	8	4
$\ln(w), (x=w+1, y=w-1)$	5	5	2

Table II. On-line delays of several CORDIC functions

The proposed on-line CORDIC algorithm attains speed improvement by introducing redundancy in the calculation circuitry. Another advantage is that on-line algorithms reduces I/O requirement and the total latency of chained processing. For example, in applications such as Givens rotation for matrix triangularization and the transformation for singular value decomposition (SVD), the angle must be computed first and then rotations are performed. Using conventional CORDIC, the total latency time of a triangularization step will be between $2.25n$ and $3n$ CORDIC steps in an implementation proposed by Ahmed et al. [18]. Ercegovac and Lang [15] have an implementation with $3n+3$ steps. However, their step cycle is considerably smaller (they estimate 2.5-4.5 times smaller), because of the use of redundant and/or on-line addition techniques. For the proposed scheme in this paper the total latency is about $n+12$ CORDIC steps. For a proper comparison with the above mentioned schemes the actual step-time must be known. This evaluation has not been done yet.

n	partition I	memory size (bits)	partition II	memory size (bits)
16	(4)	512	(2,2)	256
24	(6)	4k	(3,3)	1k
32	(8)	16k	(4,4)	2k
48	(6,6)	16k	(4,4,4)	6k
64	(8,8)	64k	(5,5,6)	24k

Table I. Memory requirements of table look-up generation of the scaling factor.

The hardware modifications as compared to the conventional CORDIC scheme are straightforward. There are two (smaller) CORDIC-like recurrence evaluators for the correction terms needed. Fig. 4 shows a possible implementation scheme. The scaling of 2^i as shown in Fig. 4 is fictual, it only indicates the relative position and does not affect the weight of the result. The critical path (i.e., the longest circuit flow propagation time) of the CORDIC implementation is either 1 shifter, 1 redundant adder, and 1 register, or 1 5-digits carry propagate adder, 1 selection, 1 redundant adder, and 1 register. Further research about the complexity of the implementation must still be carried out.

Acknowledgment we thank Tomas Lang and the referees for their helpful comments, which have greatly contributed to improve the readability of the paper.

9. References

1. M.D. Ercegovac, "An online arithmetic: an overview," SPIE Vol. 495,*Real Time Signal Processing VII*, 1984.

2. K.D. Trivedi and M.D. Ercegovac, "On-line algorithms for division and multiplication," *IEEE Trans. on Computers*, Vol. C-27, no. 7, July 1977.

3. P.K.-G. Tu, M.D. Ercegovac, "A radix-4 on-line division algorithm," *Proceedings of the 8-th Symposium on Computer Arithmetic*, Como, Italy, 1987.

4. K.D. Trivedi and J.G. Rusnak, "High radix on-line division," *Proceedings 4-th Symposium on Computer Arithmetic*, 1978.

5. V.G. Oklobdzija, M.D. Ercegovac, "An on-line square root algorithm," *IEEE Transactions on Computers*, Vol. C-31, no. 1, January 1982

6. M.J. Irwin, "A pipelined processing unit for on-line division," *Proceedings 5-th Symposium on Computer Arithmetic*, 1978.

7. R.M. Owens, "Compound algorithms for digit online arithmetic," *Proceedings 5-th Symposium on Computer Arithmetic*, 1981.

8. R.M. Owens, "Digit online algorithms for pipelined architectures," *Ph.D. Thesis*, Dept. of Computer Science, The Pennsylvania State University, 1980.

9. B.G. Delugish, "A class of algorithms for automatic evaluation of certain elementary functions in a binary computer," *Ph.D. Thesis*, Dept. of Computer Science, University of Illinois, 1975.

10. R.M. Owens and M.J. Irwin, "On-line algorithms for the design of pipelined architectures," *Proceedings of the Sixth Annual Symposium of Computer Architecture*, Philadelphia, PA, April 1979.

11. J. Volder, "The CORDIC trigonometric computing technique", *IRE Trans. Elec. Computers*, EC-8, no. 3, Sept. 1959.

12. J.S. Walther, "A unified algorithm for elementary functions," *Spring Joint Computer Conference*, 1971,

13. H.M. Ahmed, "Signal processing algorithms and architectures," *Ph.D. Dissertation*, Dept. of Electrical Engineering, Stanford University, 1982.

14. J.C. Bu, E.F.A. Deprettre and F. de Lange, "On the optimization of pipelined silicon CORDIC algorithm", Proc. EUSIPCO-86, *Signal processing III: Theories and Applications*, I.T. Young et al (Eds.), 1986.

15. M.D. Ercegovac, T. Lang, "Redundant and on-line CORDIC: application to matrix triangularization and SVD", UCLA Computer Science Department, *Techn. Report*, CSD-870046, Sept., 1987.

16. H.J. Sips, H.X. Lin, "A new model for on-line arithmetic with an application to the reciprocal calculation," *Journal of Parallel and Distributed Computing* (to appear).

17. H.X. Lin, H.J. Sips, "A novel floating-point on-line division algorithm," *Proceedings of the 8-th Symposium on Computer Arithmetic*, Como, Italy, 1987.

18. H.M. Ahmed, J.M. Delosme, and M. Morf, "Highly concurrent computing structures for matrix arithmetic and signal processing", *IEEE Computer*, Vol. 15, No. 1, Jan. 1982.

19. M.D. Ercegovac, T. Lang, "On-line scheme for computing rotation factors," *Journal of Parallel and Distributed Computing*, Academic Press, Vol. 5, no.3, June 1988.

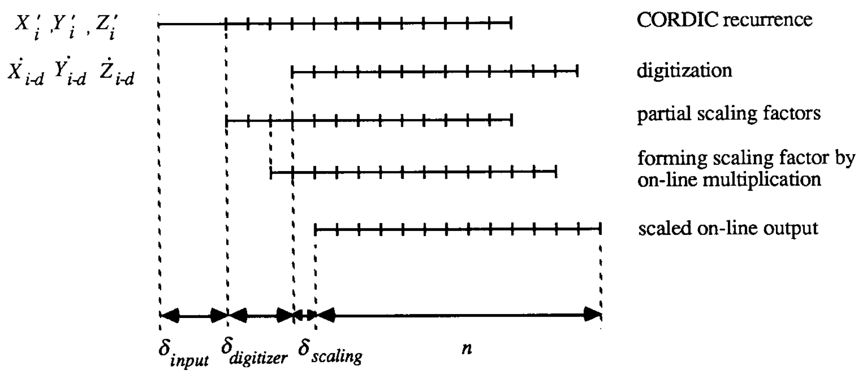


Fig.3 Illustration of timing of on-line CORDIC implementation.

Appendix A Calculation of the correction terms

In this appendix a way is shown to calculate the correction terms $\Delta X_{i+1}(x_{i+\delta}, y_{i+\delta})$ and $\Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta})$ of Eq.(4). Denote the error of X'_i and Y'_i, with respect to X_i and Y_i, as $\varepsilon_i(x)$ and $\varepsilon_i(y)$, respectively. The error in X_{i+1} and Y_{i+1} can then be written as

$$\begin{aligned} X_{i+1} &= X'_i + \varepsilon_i(x) + m\rho_i(Y'_i + \varepsilon_i(y))\cdot 2^{-i} \\ &= X'_i + m\rho_i Y'_i\cdot 2^{-i} + \varepsilon_i(x) + m\rho_i\varepsilon_i(y)\cdot 2^{-i} = X'_{i+1} + \varepsilon_{i+1}(x) \end{aligned} \tag{A1}$$

$$\begin{aligned} Y_{i+1} &= Y'_i + \varepsilon_i(y) - \rho_i(X'_i + \varepsilon_i(x))\cdot 2^{-i} \\ &= Y'_i - \rho_i X'_i\cdot 2^{-i} + \varepsilon_i(y) + \rho_i\varepsilon_i(x)\cdot 2^{-i} = Y'_{i+1} + \varepsilon_{i+1}(y) \end{aligned}$$

The term $\varepsilon_k(x)$ consists of two parts : one part depends on the digit $x_{k+\delta}$ and the other part on $y_{k+\delta}$. The same holds for $\varepsilon_k(y)$. Denote $\varepsilon_k(x)=\lambda_k(x)+\beta_k(y)$ and $\varepsilon_k(y)=\eta_k(x)+\phi_k(y)$, where $\lambda_k(x)$ and $\eta_k(x)$ are the terms depending on $x_{i+\delta}$, and $\beta_k(y)$ and $\phi_k(y)$ are the terms depending on $y_{i+\delta}$. The the following relation holds

$$\begin{aligned} \lambda_{k+1}(x) &= \lambda_k(x)+m\rho_k\eta_k(x)\cdot 2^{-k} && \text{with } \lambda_0(x)=1 \\ \beta_{k+1}(y) &= \beta_k(y)+m\rho_k\phi_k(y)\cdot 2^{-k} && \text{with } \beta_0(y)=0 \\ \eta_{k+1}(x) &= \eta_k(x)-\rho_k\lambda_k(x)\cdot 2^{-k} && \text{with } \eta_0(x)=0 \\ \phi_{k+1}(y) &= \phi_k(y)-\rho_k\beta_k(y)\cdot 2^{-k} && \text{with } \phi_0(y)=1 \end{aligned} \tag{A2}$$

The error terms can then be calculated according to

$$\begin{aligned} \varepsilon_k(x) &= \lambda_k(x)\cdot x_{i+\delta} + \beta_k(y)\cdot y_{i+\delta} = \Delta X_{i+1}(x_{i+\delta}, y_{i+\delta}) \\ \varepsilon_k(y) &= \eta_k(x)\cdot x_{i+\delta} + \phi_k(y)\cdot y_{i+\delta} = \Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta}) \end{aligned} \tag{A3}$$

The equations (A2) are two sets CORDIC like equations (Eq.(1a) and Eq.(1b)) which are to be updated each iteration cycle of the basic CORDIC equation. However, only approximately half the number of digits of the full word length are needed.

Appendix B Estimation of the Upper Bounds on ΔX_{i+1} and ΔY_{i+1}

The following relation hold for the correction terms $\Delta X_{i+1}(x_{i+\delta}, y_{i+\delta})$ and $\Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta})$,

$$\begin{aligned} \Delta X_{i+1}(x_{i+\delta}, y_{i+\delta}) &= \Delta X_i(x_{i+\delta}, y_{i+\delta}) + m\rho_i\cdot\Delta Y_i(x_{i+\delta}, y_{i+\delta})\cdot 2^{-i} \\ \Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta}) &= \Delta Y_i(x_{i+\delta}, y_{i+\delta}) - \rho_i\cdot\Delta X_i(x_{i+\delta}, y_{i+\delta})\cdot 2^{-i} \end{aligned} \tag{B1}$$

For $m=1$, with $\Delta X_0(x_{i+\delta}, y_{i+\delta})=x_{i+\delta}$ and $\Delta Y_0(x_{i+\delta}, y_{i+\delta})=y_{i+\delta}$ ($x_{i+\delta}$ and $y_{i+\delta}$ are digits with a weight of $2^{-(i+\delta+1)}$, since i starts from 0), it can be verified (e.g., simply iterating the recurrence Eq.(B1) with a computer) that

$$\begin{aligned} |\Delta X_4(x_{i+\delta}, y_{i+\delta})| &\le 2.32\cdot 2^{-(i+\delta+1)} \\ |\Delta Y_4(x_{i+\delta}, y_{i+\delta})| &\le 2.32\cdot 2^{-(i+\delta+1)} \end{aligned} \tag{B2}$$

Therefore, from Eq.(B1) and Eq.(B2) it holds,

$$\begin{aligned} |\Delta X_{i+1}(x_{i+\delta}, y_{i+\delta})| &\le 2.32\cdot 2^{-(i+\delta+1)}\cdot\prod_{j=4}^{i}(1+2^{-j}) \\ |\Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta})| &\le 2.32\cdot 2^{-(i+\delta+1)}\cdot\prod_{j=4}^{i}(1+2^{-j}) \end{aligned} \tag{B3}$$

Since

$$\begin{aligned} \prod_{j=4}^{i}(1+2^{-j}) &< \prod_{j=4}^{\infty}(1+2^{-j}) \\ &< \prod_{j=4}^{9}\left(\frac{1+2^{-j}}{1+(1/j^2\pi^2)}\right).sinh(1).\frac{1}{\prod_{j=1}^{3}(1+(1/j^2\pi^2))} \le 1.14 \end{aligned}$$

($sinh(x) = x\cdot\prod_{j=1}^{\infty}(1+x^2/(j^2\pi^2))$, and for $j\ge 10$, $(1+2^{-j})<(1+1/(j^2\pi^2))$.

it follows:

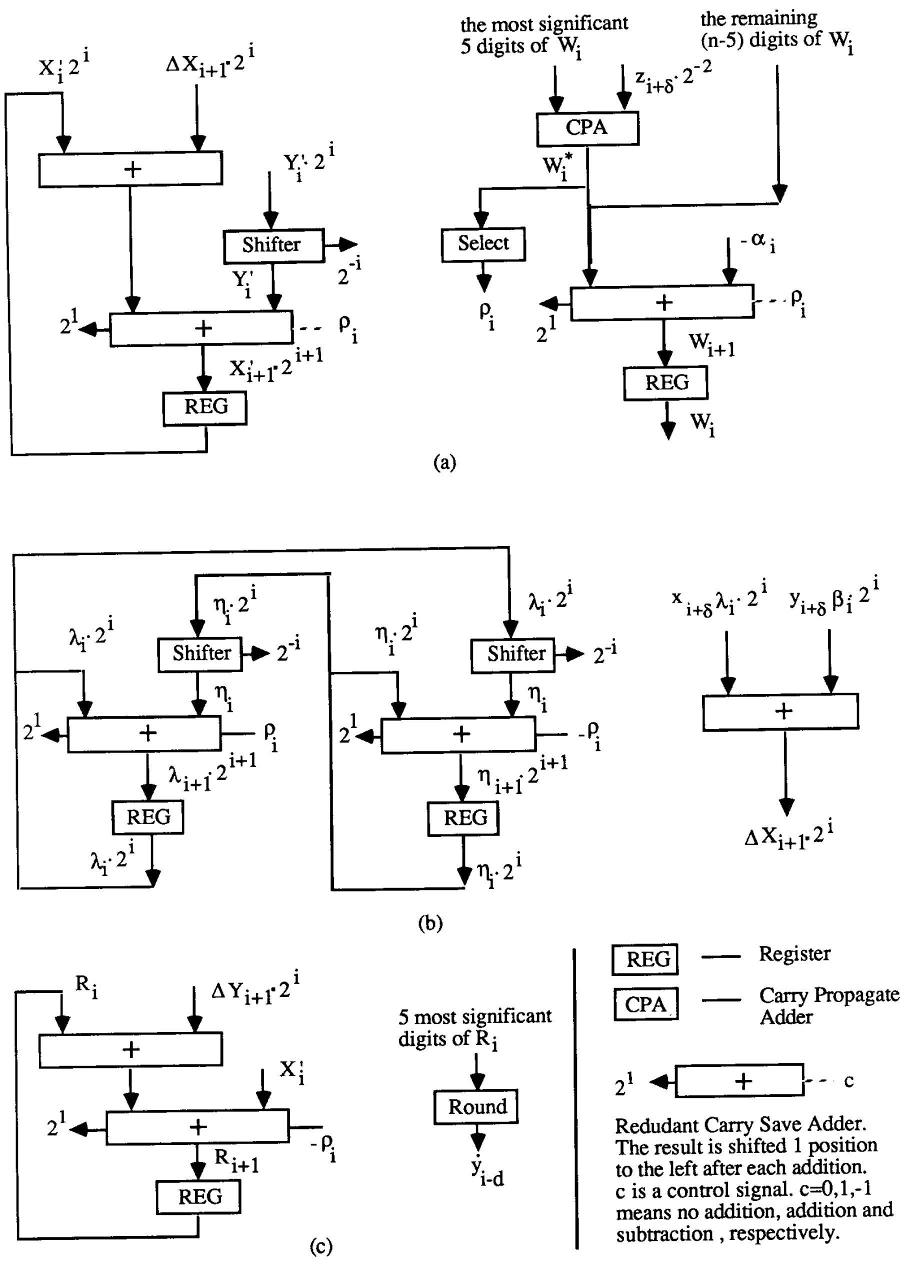
$$\begin{aligned} |\Delta X_{i+1}(x_{i+\delta}, y_{i+\delta})| &\le 2.65\cdot 2^{-(i+\delta+1)} \\ |\Delta Y_{i+1}(x_{i+\delta}, y_{i+\delta})| &\le 2.65\cdot 2^{-(i+\delta+1)} \end{aligned} \tag{B4}$$

A similar analysis can be made for

$$\sum_{k=i+\delta}^{n}[\Delta X_i(x_k,y_k).2^{-i} + \Delta Y_{i+1}(x_k,y_k)].$$

The procedure is to calculate it for several small values of i, then using Eq(B4) to estimate the upper bound. The following upper bound can be obtained,

$$\left|\sum_{k=i+\delta}^{n}[\Delta X_i(x_k,y_k)\cdot 2^{-i} + \Delta Y_{i+1}(x_k,y_k)]\right| \le 3\cdot 2^{-(i+\delta)} \quad \text{for } i\ge 0$$



$X_i \cdot 2^i$
$\Delta X_{i+1} \cdot 2^i$
$Y_i' \cdot 2^i$
Shifter
2^{-i}
Y_i'
2^1
ρ_i
$X_{i+1}' \cdot 2^{i+1}$
REG
the most significant 5 digits of W_i
the remaining (n-5) digits of W_i
$z_{i+\delta} \cdot 2^{-2}$
CPA
W_i^*
Select
ρ_i
$-\alpha_i$
W_{i+1}
W_i
(a)
$\lambda_i \cdot 2^i$
$\eta_i \cdot 2^i$
η_i
$\lambda_{i+1} \cdot 2^{i+1}$
$-\rho_i$
$\eta_{i+1} \cdot 2^{i+1}$
$x_{i+\delta} \lambda_i \cdot 2^i$
$y_{i+\delta} \beta_i \cdot 2^i$
$\Delta X_{i+1} \cdot 2^i$
(b)
R_i
$\Delta Y_{i+1} \cdot 2^i$
X_i
R_{i+1}
5 most significant digits of R_i
Round
y_{i-d}
(c)
REG — Register
CPA — Carry Propagate Adder
c

Redudant Carry Save Adder.
The result is shifted 1 position to the left after each addition.
c is a control signal. c=0,1,-1 means no addition, addition and subtraction , respectively.

Fig.4 *a. Scheme for CORDIC iterations,*
b. Scheme for correction term generation (the generation of β_i and ϕ_i are similar to that of λ_i and η_i),
c. Digitization of the iteration result.