

# **Comments on Duprat and Muller's Branching CORDIC Paper [1]**

Dhananjay S. Phatak  
Electrical Engineering Department  
State University of New York, Binghamton, NY 13902-6000

*(IEEE Transactions on Computers, vol 47, No. 9, Sep. 1998, pp 1037–1040)*

## **ABSTRACT**

**In [1], Duprat and Muller introduced the ingenious “Branching CORDIC” algorithm. It enables a fast implementation of CORDIC algorithm using signed digits and requires a constant normalization factor. This correspondence corrects some errors in the original paper. All the page numbers quoted are from [1].**

**Index Terms: Branching CORDIC, Constant Scale Factor, Signed–Digit representation, Corrections, Errata.**

The CIRCULAR ROTATION mode of CORDIC algorithm is used to evaluate Sine/Cosine/Tangent. It is based on the iteration [1]

$$X_{i+1} = X_i - s_i Y_i 2^{-i} \quad (1)$$

$$Y_{i+1} = Y_i + s_i X_i 2^{-i} \quad (2)$$

$$Z_{i+1} = Z_i - s_i \arctan 2^{-i} \quad \text{where } s_i \in \{-1, 0, +1\} \quad (3)$$

To compute  $\sin \theta_0$  and  $\cos \theta_0$ , upto  $n$  bits of accuracy,  $m = (n+2)$  iterations of the above cross-coupled equations are carried out with the the initial conditions

$$X_0 = \frac{1}{K} \quad (4)$$

$$Y_0 = 0 \quad (5)$$

$$Z_0 = \theta_0 \quad (6)$$

At each (say  $i$ th) step,  $s_i$ ,  $i = 0, \dots, m-1$ , are selected so that

$$Z_{i+1} \longrightarrow 0 \quad \text{or} \quad (7)$$

$$\sum_{i=0}^{m-1} \arctan(s_i 2^{-i}) \longrightarrow \theta_0 \quad (8)$$

Correspondingly, the value of  $K$  is

$$K = \prod_{i=0}^{m-1} K_i = \prod_{i=0}^{m-1} \sqrt{1 + (s_i 2^{-i})^2} \quad (9)$$

In general, the coefficients  $s_i$  at each step of the CORDIC iteration can take any of the three values  $\{-1, 0, +1\}$ . If  $s_i = 0$  is allowed, then the scaling factor  $K$  is not a constant, but depends on the actual sequence of  $s_i$  values. On the other hand, if  $s_i$  can be restricted to  $\pm 1$ , then  $K$  is a constant (since the number of iterations  $m$  that are to be executed for a given precision are known ahead of time). For this method to work, the initial angle must satisfy

$$|Z_0| \leq \sum_{k=0}^{\infty} \arctan 2^{-k} = 1.74328662 \quad (10)$$

This range covers all angles of practical interest since  $\frac{\pi}{2} \approx 1.5708 < \sum_{k=0}^{\infty} \arctan 2^{-k}$ .

With the introduction of (Redundant) Signed-Digit representations [2, 3, 4] the addition becomes carry-free, (i.e., the addition takes a small fixed amount of time, irrespective of the wordlength) thus offering a potential for significant speedup. To fully exploit the speed advantage gained by using signed digits, the sign detection of the residual angle also must be done in a constant (and small) time delay (note that the next action depends on whether the current residual angle is positive or negative). This in turn implies that only a fixed number of leading digits can be looked at to determine the sign of the residual angle. In most methods (for example, those in [1, 5]) a window of 3 (leading) digits turns out to be sufficient to determine the sign. At each iteration, the window shifts right by one digit position. If at least one of the digits in the window of interest is non-zero, the sign of the residual angle can be determined to be  $\pm 1$ . If the sign is  $+1$ , the next elementary angle ( $\arctan 2^{-i}$  at step  $i$ ) should be subtracted, if the sign is  $-1$ , the next elementary angle should be added. The problem occurs when all the digits in the window of interest are zero or in other words the residual angle has many leading zeroes, so that just by looking at the window of 3 (leading) digits, it is not possible to tell whether its sign is  $+1$  or  $-1$ . Ideally, in this case, one should select  $s_i = 0$  and neither add nor subtract the elemental angle for that step. However, the coefficients  $s_i$  must be restricted to  $\{-1, +1\}$  to render the scaling factor  $K$  to be a constant.

Duprat and Muller's Branching CORDIC algorithm [1] circumvents this difficulty by initiating two separate CORDIC rotations in parallel: one assuming  $s_i = +1$  and one assuming  $s_i = -1$ . It might appear that this branching could in turn lead to further branchings down the line. The ingenuity of their

method essentially lies in realizing that further branchings are not possible and that a branching either terminates eventually, or if it does not terminate till the end, then *both* the modules have the correct result (within the tolerance specified). In summary, their method employs two separate modules to implement the iteration in the zeroing part (i.e., equation (3)). Whenever the sign of the residual angle can  $Z_i$  can be unambiguously determined, both modules do identical operations. Otherwise the algorithm enters a branching.

We have recently shown that this algorithm can be enhanced to perform *two rotations* in a single step (i.e., use two elementary angles in each module at every step) [6]. In our “Double Step Branching CORDIC” algorithm, every module performs distinct operations in each step (irrespective of whether a branching is on-going), leading to better hardware utilization. The double stepping method could also lead to speed enhancement (relative to Duprat and Muller’s original Branching CORDIC) depending on the actual VLSI implementation. During the analytical proof and independent experimental verification (via extensive simulations) of the Double Step Branching CORDIC method, some errors were found in Duprat and Muller’s original paper [1]. This correspondence corrects those errors.

In [1] the two modules used to implement the “zeroing iteration” (equation (3)) are called “+” and “-” modules. Variables (input and output residual angles, etc.) associated with the two modules are designated by superscripts “+” and “-” respectively. In the notation of their paper, at step  $i$ , the modules generate

$$\text{Module } + : z_{i+1}^+ = z_i^+ - d_i^+ \arctan 2^{-i} \quad \text{where } d_i^+ = \pm 1 \quad (11)$$

$$\text{Module } - : z_{i+1}^- = z_i^- - d_i^- \arctan 2^{-i} \quad \text{where } d_i^- = \pm 1 \quad (12)$$

Prior to step  $i$ , elementary angles  $\{\arctan 2^{-0}, \dots, \arctan 2^{-(i-1)}\}$  have already been used, and at step  $i$ , the modules perform appropriate operation (addition or subtraction) using the next elementary angle ( $\arctan 2^{-i}$ ) to further reduce the magnitude of the next residues  $z_{i+1}^+$  and  $z_{i+1}^-$ . Both modules then detect the sign of the residual angles  $z_{i+1}^+$  and  $z_{i+1}^-$  in parallel, by looking at a window of 3 digits.

The output of the sign detection operation performed on a residual angle  $z_i$  is denoted by “eval( $z_i$ )” in their paper. eval( $z_i$ ) can assume any of the 3 values  $\{-1, 0, +1\}$ . When all the 3 digits in the window of interest are 0, the “eval” function returns a 0, otherwise eval returns  $\pm 1$ . If eval( $z_i$ )=1, then  $z_i > 0$ . Similarly, if eval( $z_i$ )= -1, then  $z_i < 0$ . If eval( $z_i$ )=0, there is insufficient information to determine whether  $z_i$  is positive or negative and their algorithm enters a “branching”.

When the Branching CORDIC algorithm described in [1] is followed, it turns out that at the beginning of step  $i$ , (i.e., having used elementary angles upto and including  $\arctan 2^{-(i-1)}$  and prior to using the elementary angle  $\arctan 2^{-i}$ ), the residues  $z_i^+$  and  $z_i^-$  satisfy both of the following bounds:

$$\text{the “tighter” bound :} \quad \text{at least one of } |z_i^+| \text{ and } |z_i^-| \text{ is } \leq \sum_{k=i}^{\infty} \arctan 2^{-k} \quad (13)$$

$$\text{the “coarser” bound :} \quad \text{both } |z_i^+| \text{ and } |z_i^-| \text{ are } \leq 3 \cdot 2^{-(i-1)} \quad (14)$$

The two bounds above are labeled “tighter” and “coarser” in this correspondence because

$$\sum_{k=i}^{\infty} \arctan 2^{-k} < 2^{-(i-1)} < 3 \cdot 2^{-(i-1)} \quad (15)$$

The statement of Theorem 1 on page 171 in [1] states the above “tighter” bound and “coarser” bound which the residual angles must satisfy at each step.

It can be verified that the “*coarser*” bound implies that 3 digits of weight  $[2^{-(i-3)}, 2^{-(i-2)}, 2^{-(i-1)}]$  must be examined to evaluate the sign of  $z_i$ , prior to using  $\arctan 2^{-i}$ .

However, in the body of the proof of Theorem 1 in [1] and in the following section (in particular in Table III on page 173 in [1]), the text suggests that a window of 3 digits spanning positions of weight  $[2^{-(i-1)}, 2^{-i}, 2^{-(i+1)}]$  is to be used when determining the sign of  $z_i$ , which is incorrect. The correct and incorrect windows are pictorially illustrated in Figure 1.

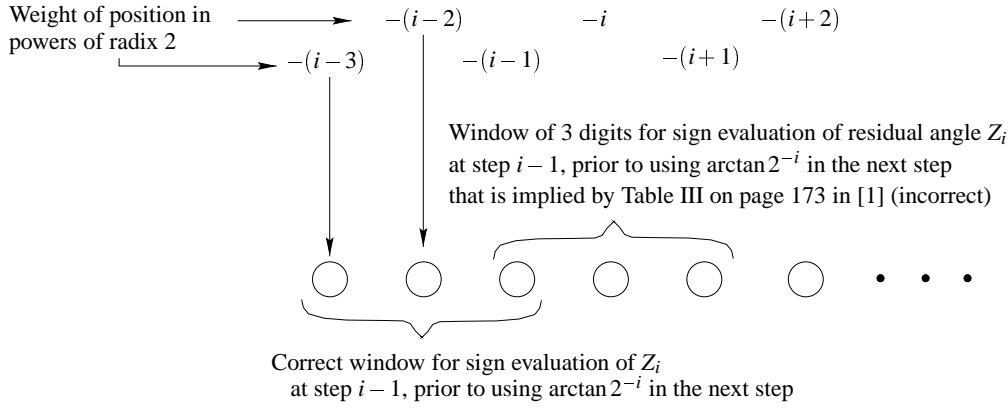


Figure 1: The window of 3 leading digits implied in [1] which is used to evaluate sign of residual angle  $z_i$  prior to using  $\arctan 2^{-i}$  in the next step (incorrect). The correct window position is shifted by two positions to the left as shown.

That the correct window spans digit positions  $[2^{-(i-3)}, 2^{-(i-2)}, 2^{-(i-1)}]$  has been verified through analytical derivations as well as via simulations. Duprat and Muller’s algorithm was simulated with 4 different windows (for sign evaluation prior to using  $\arctan 2^{-i}$ )

- (W1)  $[2^{-(i-4)}, 2^{-(i-3)}, 2^{-(i-2)}]$
- (W2)  $[2^{-(i-3)}, 2^{-(i-2)}, 2^{-(i-1)}]$
- (W3)  $[2^{-(i-2)}, 2^{-(i-1)}, 2^{-i}]$
- (W4)  $[2^{-(i-1)}, 2^{-i}, 2^{-(i+1)}]$

Only window (W2) yields correct results as required by the “*coarser*” bound . All other window positions lead to residual angles that violate the “*coarser*” bound and hence to incorrect results.

Given the correct window above,  
if  $\text{eval}(z_i) = 0$  then  $|z_i| < 2^{-(i-1)}$  since all leading digits up to position  $2^{-(i-1)}$  are 0 (16)

The original paper [1] erroneously assumes that  
if  $\text{eval}(z_i) = 0$  then  $|z_i| < 2^{-(i+1)}$   
because of the incorrect window position. Consequently, some parts of the original proof need to be modified.

In particular, the proof of the case labeled  
“ 2) If  $s^+ = 0$  then (the case  $s^- = 0$  is symmetrical)  $\dots$  ”  
(on page 172 in [1] near the bottom of the first column) and some of the following material needs to  
be modified as indicated below.

Without loss of generality, assume that the current branching starts at step  $i$ , i.e.,  
 $\text{eval}(z_i) = 0$ , so that the modules perform

$$\text{Module } + : z_{i+1}^+ = z_i - \arctan 2^{-i} \quad (17)$$

$$\text{Module } - : z_{i+1}^- = z_i + \arctan 2^{-i} \quad \text{so that} \quad (18)$$

$$z_{i+1}^- - z_{i+1}^+ = 2 \arctan 2^{-i} \quad (19)$$

If the branching continues at step  $i + 1$  then

$$\text{eval}(z_{i+1}^+) = -1 \quad \text{and} \quad \text{eval}(z_{i+1}^-) = +1 \quad \text{which implies that } z_{i+1}^+ < 0 \quad \text{and} \quad z_{i+1}^- > 0, \quad \text{or} \quad (20)$$

$$|z_{i+1}^-| + |z_{i+1}^+| = 2 \arctan 2^{-i} \quad (21)$$

The last equation in turn implies that at least one of  $|z_{i+1}^-|$  and  $|z_{i+1}^+|$  is less than or equal to  $\arctan 2^{-i} \leq \sum_{k=i+1}^{\infty} \arctan 2^{-k}$ , or in other words, at least one of  $|z_{i+1}^-|$  and  $|z_{i+1}^+|$  satisfies the “*tighter*” bound as required. From this step onwards, if the branching continues till step  $p$  then the “+” module keeps subtracting while the “-” module keeps adding the subsequent angles. As a result, it is easy to show that [6] whichever module returns a residue sign value different from that of the previous residue has the correct output, i.e., it’s output satisfies the “*tighter*” bound .

The problem occurs when the branching which started at step  $i$  stops at step  $i + 1$  with one of the modules returning a “0” sign. For example, if  $\text{eval}(z_{i+1}^+) = 0$ , then  $z_{i+1}^+$  is deemed to be the correct output. Hence we must show that  $z_{i+1}^+$  satisfies the “*tighter*” bound .

Let  $LZ(y)$  denote the leading zeroes in a signed digit number  $y$ . Then  
 $\text{eval}(z_i) = 0$  implies  $LZ(z_i) = i - 1$  and vice versa. (22)

With this notation, we use the following result:

**Lemma 1 :** If  $\text{eval}(z_i) = 0$ , i.e.,  $LZ(z_i) = i - 1$  and  $z_{i+1}^+$  and  $z_{i+1}^-$  are generated as per equations (17) and (18), then

(i) if  $\text{eval}(z_{i+1}^+) = 0$ , i.e., if  $LZ(z_{i+1}^+) = i$  then  $z_i > 0$

(ii) if  $\text{eval}(z_{i+1}^-) = 0$ , i.e., if  $LZ(z_{i+1}^-) = i$  then  $z_i < 0$

**Proof :** We show the proof of part (i) only since the other part can be proved identically. Note that  $\arctan 2^{-i}$  has zeroes in all digit positions up to  $2^{-(i-1)}$  for all values of  $i$ . The leading “1” of the  $n$  bit number that is closest to  $\arctan 2^{-i}$  is in position  $2^{-(i+1)}$  for  $i < \frac{n}{3}$  and in position  $2^{-i}$  for  $i \geq \frac{n}{3}$  where  $n$  is the target precision. Correspondingly the digits of  $\arctan 2^{-i}$  in positions  $[2^{-i}, 2^{-(i+1)}, 2^{-(i+2)}]$  are [0 1 1] or [1 0 0], respectively. We illustrate the proof assuming only the former (the other case can be proved identically). For this case the digit patterns for residue  $z_i$ ,  $\arctan 2^{-i}$  and the next residue  $z_{i+1}^+$  are as shown in Figure 2.

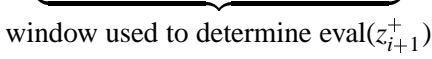
digit positions of interest $\rightarrow$	$2^{-(i-2)}$	$2^{-(i-1)}$	$2^{-i}$	$2^{-(i+1)}$	$2^{-(i+2)}$
digit pattern of $z_i \rightarrow$ $d_k \in \{-1, 0, +1\}, k \geq i$	0	0	$d_i$	$d_{i+1}$	$d_{i+2}$
leading digits of $(-\arctan 2^{-i}) \rightarrow$			0	-1	-1
<hr/>					
leading digits of $z_{i+1}^+ \rightarrow$ $s_k \in \{-1, 0, +1\}, k \geq i+1$	0	0	$s_i = 0$	$s_{i+1}$	$s_{i+2}$
					

Figure 2: Illustration of digit patterns for  $z_i$ ,  $\arctan 2^{-i}$  and the next residue  $z_{i+1}^+ = z_i - \arctan 2^{-i}$ .

The operands are signed digits and hence rules for adding radix-2 signed digits are used to perform the addition (these can be found in [2], [4] and [7]). For our purpose, it suffices to know that the sum is carried out in two steps. In the first step, an intermediate sum and an intermediate carry-out are generated at all digit positions (in parallel). In the next step, the intermediate sum and the intermediate carry-in (from adjacent lower significant digit position) are added to generate the final sum output. The intermediate carry is selected depending only on the digit sums in the current and adjacent lower significant positions in such a way that the final summation in the second step does not generate a carry. In essence, redundancy available in the signed digit representation is exploited to stop carry propagation.

In the above figure we need to show that  $z_i > 0$ , i.e.,

- (i) at least one of the digits  $\{d_i, d_{i+1}, d_{i+2}\}$  is non-zero and
- (ii) the first nonzero digit is a +1

which is proved as follows:

If  $d_i = 1$  then  $z_i > 0$  and the proof is complete. If  $d_i = -1$  then the digit sum (of the two operand digits) at position  $2^{-i}$  is  $-1$ . Since the digit sum in the adjacent position  $2^{-(i+1)}$  is  $(d_{i+1} - 1) \leq 0$ , the carry from position  $2^{-(i+1)}$  can have only one of the two values  $\{-1, 0\}$ . Consequently, the intermediate carry-out of position  $2^i$  must be  $-1$ . This would imply that the digit in  $2^{-(i-1)}$ th position of  $z_{i+1}^+$  would be  $-1$ , which contradicts the given condition, viz.,  $LZ(z_{i+1}^+) = i$ . Hence  $d_i \neq -1$ . That leaves the possibility  $d_i = 0$ . In this case, examine the adjacent digit  $d_{i+1}$ . If this digit is  $+1$ , then  $z_i > 0$  and the proof is done. If  $d_{i+1} = -1$  (along with  $d_i = 0$ ) then the intermediate carry-out of digit position  $2^{-(i+1)}$  must be  $-1$  (since the digit sum at that position is  $-2$ ). This implies a final sum digit  $s_i = -1$  which contradicts the condition  $LZ(z_{i+1}^+) = i$  and consequently  $d_{i+1} \neq -1$ . That leaves the possibility that  $d_{i+1} = 0$  (along with  $d_i = 0$ ). In this case, examine  $d_{i+2}$ . If it is  $+1$ , then we are done. If it is  $0$  or  $-1$  then the digit sums at positions  $2^{-(i+1)}$  as well as  $2^{-(i+2)}$  are strictly negative. Hence, the carry-out of position  $2^{-(i+1)}$  is  $-1$  which implies a final sum digit  $s_i = -1$  which contradicts the condition  $LZ(z_{i+1}^+) = i$  and consequently  $d_{i+2}$  cannot be  $0$  or  $-1$ . Thus, we have proved that the leading digit of  $z_i$  must be a  $+1$  which implies that  $z_i > 0$ , completing the proof of Lemma 1.

With the help of Lemma 1; the proof for the case when branching starts at step  $i$  and stops at step  $i + 1$  with one of the modules returning a zero value for the evaluated sign, can be completed as follows.

Condition (i) of Lemma 1, along with equation (17) implies that

$z_{i+1}^+ = z_i - \arctan 2^{-i} > -\arctan 2^{-i}$ . Since  $z_i$  satisfied the “tighter” bound, it follows that  $-\arctan 2^{-i} < z_{i+1}^+ \leq \sum_{k=i+1}^{\infty} \arctan 2^{-k}$ , i.e.  $|z_{i+1}^+|$  satisfies the “tighter” bound as required.

Similarly, condition (ii) of Lemma 1, along with equation (18) implies that

$z_{i+1}^- = z_i + \arctan 2^{-i} < \arctan 2^{-i}$ . Since  $z_i$  satisfied the “tighter” bound, it follows that  $-\sum_{k=i+1}^{\infty} \arctan 2^{-k} \leq z_{i+1}^- < \arctan 2^{-i}$ , i.e.  $|z_{i+1}^-|$  satisfies the “tighter” bound as required.

This completes the corrections to the proofs in [1].

## Acknowledgment

The author would like to thank Dr. Jean-Michel Muller for a quick and thorough review of this manuscript. He corrected some errors in this correspondence and his constructive comments helped improve the quality of this submission. The corrected version of the algorithm appears in his exhaustive new book on elementary function evaluations [8] (along with all kinds of improvements on and variations of the basic CORDIC algorithm).

## References

- [1] J. Duprat and J. Muller, “The CORDIC algorithm: new results for fast VLSI implementation,” *IEEE Trans. on Computers*, vol. TC-42, pp. 168–178, Feb. 1993.
- [2] I. Koren, *Computer Arithmetic Algorithms*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1993.
- [3] B. Parhami, “Generalized signed-digit number systems: a unifying framework for redundant number representations,” *IEEE Transactions on Computers*, vol. C-39, pp. 89–98, Jan. 1990.
- [4] D. S. Phatak and I. Koren, “Hybrid Signed-Digit Number Systems: A Unified Framework for Redundant Number Representations with Bounded Carry Propagation Chains,” *IEEE Trans. on Computers, Special issue on Computer Arithmetic*, vol. TC-43, pp. 880–891, Aug. 1994. (An unabridged version is available on the web via the URL <http://www.ee.binghamton.edu/faculty/phatak>).
- [5] N. Takagi, T. Asada, and S. Yajima, “Redundant CORDIC methods with a constant scale factor for Sine and Cosine computation,” *IEEE Trans. on Computers*, vol. 40, pp. 989–999, Sep. 1991.
- [6] D. S. Phatak, “Double Step Branching CORDIC: A New Algorithm for Fast Sine and Cosine Generation,” *IEEE Transactions on Computers*, vol. TC-47, pp. 587–602, May 1998.
- [7] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, “A high-Speed multiplier using a redundant binary adder tree,” *IEEE Journal of Solid-State Circuits*, vol. SC-22, pp. 28–34, Feb. 1987.
- [8] J. Muller, *Elementary Functions, Algorithms and Implementation*. Birkhauser Publishers, Boston, 1997.