Evaluation of Complex Elementary Functions A New Version of BKM

Jean-Claude Bajard and Laurent Imbert

LIM – CNRS ESA 6077
Université de Provence, Marseille, France

ABSTRACT

We present an improvement of BKM: a shift-and-add algorithm based on the CORDIC that allows fast computation of complex exponential and logarithm functions. BKM is accelerated by the use of a redundant binary number system. Unlike the previous redundant CORDIC methods, we do not need neither to calculate the scale factor during the computation, nor to double the number of iterations. So our algorithm is suitable for an efficient hardware implementation.

Keywords: BKM, elementary functions, redundant number system, CORDIC, complex numbers, hardware algorithm.

1. INTRODUCTION

The CORDIC algorithm was introduced by J. Volder [1] in 1959 and generalized by Walther [2] in 1971 to compute logarithms, exponentials, trigonometric functions, rotations and square roots. It is very attractive since the same algorithm allows the computation of many elementary functions using only simple shift-and-add steps. CORDIC is implemented in many pocket calculators and arithmetic coprocessors. Recent survey can be found in [3]. In an other hand, Henry Briggs had proposed in the 17th century an algorithm for computing logarithms which was used to make the first logarithm table.

The BKM algorithm which was presented for the first time in [4], merges CORDIC and Briggs' algorithm. It allows us to compute many complex functions (logarithm, exponential, ...) and to perform rotations, as well as complex multiplications and divisions without scaling factor. BKM is very attractive for an hardware implementation since it takes both advantages of the shifts-and-add methods, and in the use of redundant arithmetic [5].

Nevertheless, in its original form, the computation of the complex logarithm (L-mode of BKM) is not trivial since it requires a dedicated first step with many comparisons. In this paper, we propose some improvements for the computation of the complex logarithm. This new algorithm converges in a slightly larger domain. We define a new range reduction step which can be easily performed by additions and shifts, and we propose a new approach to the proof of the convergence.

In the next section we will describe the former BKM algorithm. Then, in section 3 we propose our improvements of the L-mode and proved it in section 4. A dedicated circuit will be proposed in section 6.

2. THE BKM ITERATION

The BKM algorithm computes the complex exponential and logarithm functions. From these computations, we can deduce a lot of other complex or real functions such as trigonometric functions, 2-D rotations, complex multiplications and divisions, square roots, ...

As CORDIC based methods, BKM essentially consists of finding a representation of a number in a numeric system, which is interpreted in another one to obtain the desired evaluation [6].

Further author information: (Send correspondence to Laurent Imbert)
Jean-Claude Bajard : Jean-Claude.Bajard@gyptis.univ-mrs.fr
Laurent Imbert : Laurent.Imbert@gyptis.univ-mrs.fr

For instance, to compute the logarithm of x we find a sequence $d_k = -1, 0, 1$, such that:

$$x \prod_{k=1}^{n} \left(1 + d_k 2^{-k}\right) \approx 1 \tag{1}$$

Thus, the representation of $\frac{1}{x}$ in this multiplicative system gives the logarithm of x in the additive one.

$$\ln x \approx -\sum_{k=1}^{n} \ln\left(1 + d_k 2^{-k}\right) \tag{2}$$

BKM is based on the following iterations:

$$E_{k+1} = E_k \left(1 + d_k 2^{-k}\right) \tag{3}$$

$$L_{k+1} = L_k - \ln\left(1 + d_k 2^{-k}\right) \tag{4}$$

$$\text{with } d_k = d_k^r + i\, d_k^i, \text{ and } d_k^r, d_k^i = -1,\ 0,\ 1$$

- **E-mode**: If we find a sequence d_k such that L_k goes to 0, then we will obtain $E_k \rightarrow E_1 e^{L_1}$.
- **L-mode**: If we find a sequence d_k such that E_k goes to 1, then we will obtain $L_k \rightarrow L_1 + \ln(E_1)$.

In the following we will focus on the L-mode of the algorithm since it is the most complex part of the original algorithm and it needed to be improved. The E-mode is well studied in [4].

3. COMPUTATION OF THE LOGARITHM FUNCTION (L-MODE)

So, in this mode, our goal is to find a sequence d_k such that E_k goes to 1.

If we define: $S_k = 2^k (E_k - 1)$, then eq. 3 becomes:

$$S_{k+1} = 2(S_k + d_k) + 2^{-k+1} S_k d_k \tag{5}$$

One can easily see that if we find a sequence d_k such that S_k is bounded, then E_k converges to 1. We propose the following algorithm to computes such a sequence.

3.1. New BKM algorithm for the L-mode:

- Start with $E_1 \in P_0 = \{x + iy\,;\ 0.64 \leq x \leq 1.4,\ and\ -\frac{2}{5}x \leq y \leq \frac{2}{5}x\}$
- Initialize $S_2 = 2^2 (E_1 - 1)$
- for $k = 2$ to n (n depends on the accuracy required)
  - $\begin{cases} S_{k+1} = 2(S_k + d_k) + 2^{-k+1} S_k d_k \\ L_{k+1} = L_k - \ln\left(1 + d_k 2^{-k}\right) \end{cases}$
    with $d_k = d_k^r + i d_k^i$, and $d_k^r,\ d_k^i = -1, 0, 1$.
  - define $\widetilde{S}_k^r$ et $\widetilde{S}_k^i$ as the values obtained by truncating the real and imaginary parts of S_k^r and S_k^i after their 4th fractional digits, where $S_k = S_k^r + i S_k^i$.
  - $\begin{cases} if\ \widetilde{S}_k^r \leq -\frac{1}{2} & then,\ d_k^r = 1 \\ if\ -\frac{1}{2} < \widetilde{S}_k^r \leq \frac{1}{2} & then,\ d_k^r = 0 \\ if\ \frac{1}{2} < \widetilde{S}_k^r & then,\ d_k^r = \overline{1} \end{cases}$
  - $\begin{cases} if\ \widetilde{S}_k^i \leq -\frac{1}{2} & then,\ d_k^i = 1 \\ if\ -\frac{1}{2} < \widetilde{S}_k^i \leq \frac{1}{2} & then,\ d_k^i = 0 \\ if\ \frac{1}{2} < \widetilde{S}_k^i & then,\ d_k^i = \overline{1} \end{cases}$
- Result: $\begin{cases} S_k\ bounded \\ L_k \longrightarrow L_1 + \ln(E_1) \end{cases}$

Remarks:

The major differences with the algorithm presented in [4] can be resumed in two points:

- The convergence domain is slightly different: $E_1 \in P_0 = \{x + iy\,;\, 0.64 \leq x \leq 1.4,\ and\ -\frac{2}{5}x \leq y \leq \frac{2}{5}x\}$
- The first iteration of the original algorithm, which was rather tricky, is now useless since this new domain allows us to choose $d_1 = 0$. Thus we have only one kind of very simple iteration, and then, the algorithm proposed is now really efficient for an hardware implementation.

3.2. About the range reduction

The modification of the convergence domain modifies also the reduction part of the algorithm. For the logarithm function, the range reduction is multiplicative. Thus, in order to compute the logarithm of z, we have to define K such that $K \times z$ becomes to the convergence domain of the algorithm, then compute $\ln(z) = \ln(K \times z) - \ln(K)$. K is selected to reduce the multiplication $K \times z$ to shifts and additions.

For our algorithm the range reduction is very easy since we can define K with relatively simple comparisons. Our goal is to define a constant K such that, for each complex $z = x + iy$, the product $K \times z \in P_0 = \{x + iy\,;\, 0.64 \leq x \leq 1.4,\ and\ -\frac{2}{5}x \leq y \leq \frac{2}{5}x\}$.

The range reduction is done in 3 steps.

In the first one we use symmetrical properties to obtain a point $z = x + iy$ belonging to the domain:

$$\begin{cases} x \geq 0 \\ 0 \leq y \leq x \end{cases}$$

if we consider the polar representation of $z = \rho e^{i\theta}$, this square with $0 \leq \theta \leq \frac{\pi}{4}$.

Then we define the complex number k as follow:

$$\begin{cases} if & x \leq y < \frac{x}{2} & then & k = 1 - i \\ if & \frac{x}{2} \leq y < \frac{x}{4} & then & k = 1 - \frac{i}{2} \end{cases}$$

Thus, we obtain a point $z' = k \times z$ such that $-\frac{\Re(z')}{3} \leq \Im(z') \leq 0$. This ensure that z belongs to an area which is included into the convergence domain between $x = 0.64$ and $x = 1.4$.

In the third step, we define the real number k' as the power of 2 which satisfies $0.64 \leq \Re(k' \times z') \leq 1.4$. The figure 1 shows the 3 steps of this reduction step.

One can notice that the three steps requires multiplications but each can be done using only additions and shifts. Thus, that new range reduction is very easy to perform.

4. A NEW PROOF OF THE CONVERGENCE

In [4], we used the following property: "if $n \geq 4$ and $\|S_n\| \leq \frac{3}{2}$, then for any $k > n$, $\|S_k\| \leq \frac{3}{2}$", and we proved that for $E_1 \in \left\{x + iy\,;\, 0.5 \leq x \leq 1.3\ and\ -\frac{x}{2} \leq y \leq \frac{x}{2}\right\}$, the algorithm builds a sequence S_k such that for $k = 6$, $\|S_6\| \leq \frac{3}{2}$. Thus, considering the previous property, we obtained that for $k \geq 6$, $\|S_k\| \leq \frac{3}{2}$, in other words, that the algorithm converges. The major drawback of that proof is the number computations. Actually, the original proof required a program which computes approximately 9^6 vertex of polygons.

Now, we propose a new method to the proof which requires very much less computations. Thus we can perform it by a pen-and-pencil method. As the original one, this new proof is geometric and uses the same property: "if $n \geq 4$ and $\|S_n\| \leq \frac{3}{2}$, then for any $k > n$, $\|S_k\| \leq \frac{3}{2}$". In the following, we will consider complex numbers as 2-D points, and operations as plane transformations.

Each iteration of the algorithm computes the transformation τ_{d_k} , where the value S_{k+1} is the image of S_k:

$$\tau_{d_k} \,:\, S_k \longmapsto S_{k+1} = 2S_k\left(1 + d_k 2^{-k}\right) + 2d_k$$

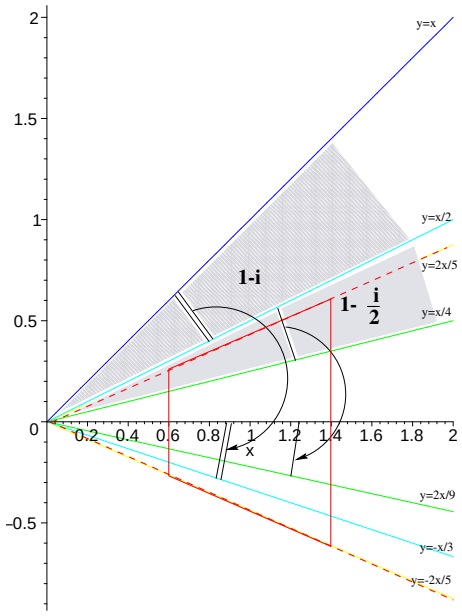


Figure 1. Range reduction for the logarithm function (L-mode).

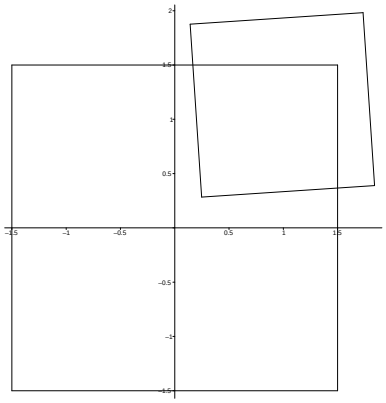


Figure 2. the square P, and its antecedent computed with $d_k = -1 - i$.

Unlike the previous one, that new proof is ascendant. We consider one step $k+1 \geq 4$, and we compute the values at step k. Let us consider that for $k+1 \geq 4$, $\|S_{k+1}\| \leq \frac{3}{2}$, in other words, S_{k+1} is inside the square $P = \left[-\frac{3}{2}, \frac{3}{2}\right] + i\left[-\frac{3}{2}, \frac{3}{2}\right]$. In order to find antecedents of S_{k+1}, we have to compute the 9 antecedents of P depending on the 9 values of d_k $(-1-i,\ -1,\ -1+i,\ -i,\ 0,\ i,\ 1-i,\ 1,\ 1+i)$.

We define the reciprocal function:

$$\tau_{d_k}^{-1} \ : \ S_{k+1} \longmapsto S_k = \frac{S_{k+1} - 2d_k}{2\left(1 + d_k 2^{-k}\right)}$$

For instance, figure 2 shows P and one of its antecedent computed with $d_k = -1-i$. If we call P' that antecedent, it is clear that the image, by τ_{-1-i}, of each point of P' belongs to P.

The principle of our new proof is to start with the square P at one step greater than 4, and to compute all its antecedents for the previous step. Then go back like that up to the first iteration.

To be sure to overlap our convergence domain, we start with P at step 7 and we compute its 9 antecedent at step 6. We obtain 9 overlapping polygons. The union of these polygons could be used to go back to step 5, but in order

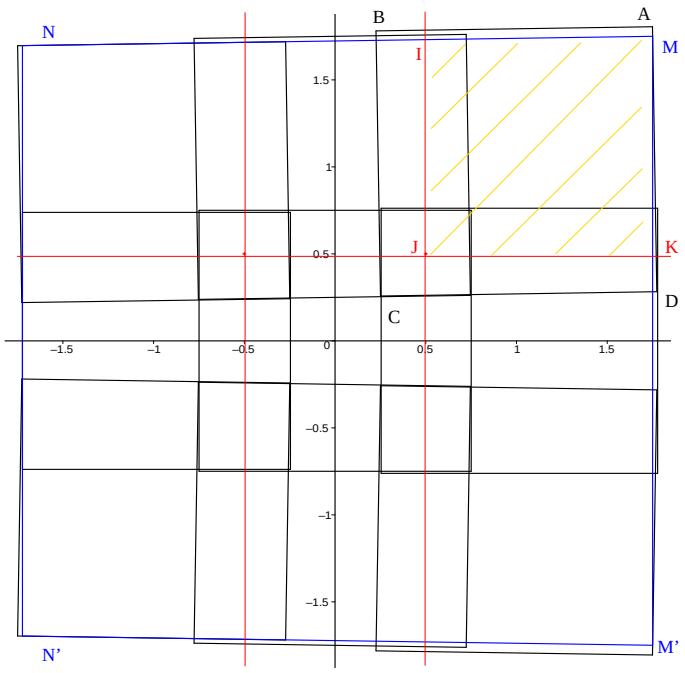


Figure 3. The trapezoid T_6 defined with the 9 antecedents of P at step 6.

to decrease the number of computations, we define a trapezoid enclosed into this union of 9 polygons stemmed from P (see figure 3). In the following, we call T_k the trapezoid defined at step k.

Now we must verify that the trapezoid T_6 is correct. If we apply the algorithm to each points of T_6, whatever the choice of the algorithm may be (see dashed lines on fig 3), the points computed must be in the square P at step 7. Let us explain that with an example:

Let $(A,\, B,\, C,\, D)$ be the antecedent of P computed with $d_6 = -1 - i$.
$T_6 = (M,\, N,\, N',\, M')$ is the trapezoid defined using the 9 antecedents of P at step 6. I is the point of intersection of the segment $[MN]$ and the straight line $x = \frac{1}{2}$, K is the point of intersection of the segment $[MM']$ and the straight line $y = \frac{1}{2}$, and J is the point of coordinates $\left(\frac{1}{2}, \frac{1}{2}\right)$. The two straight lines correspond to the splitting into which the algorithm gives $d_6 = -1 - i$.
It is easy to verify that the points M, I, J, and K belongs to the polygon $(A,\, B,\, C,\, D)$, and therefore the convex polygon $(M,\, I,\, J,\, K)$ is enclosed into the polygon $(A,\, B,\, C,\, D)$.
Thus we verify that the choice of $d_6 = -1 - i$, for the points of the polygon $(M,\, I,\, J,\, K)$ allows to embed it in P.

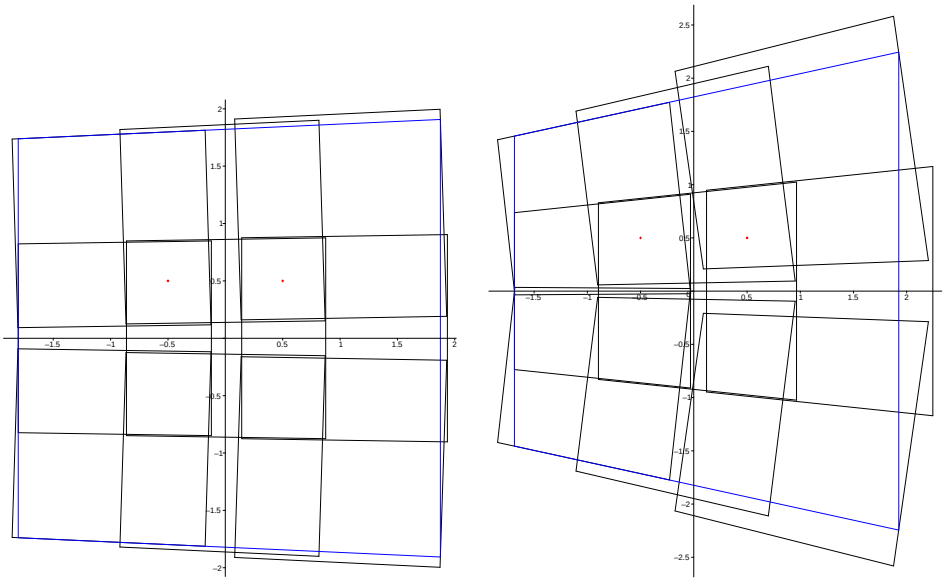
As we use redundant number system to write S_k and L_k, we can not easily make comparisons, so we only examine the first $4th$ digits of the real and imaginary parts of S_k to choose each d_k. Thus, in this proof, we must take care to the error due to the comparisons. For instance, rather than consider $x = \frac{1}{2}$ and $y = \frac{1}{2}$ we use the two straight lines $x = \frac{1}{2} - \frac{1}{16}$ and $y = \frac{1}{2} - \frac{1}{16}$ in our example. We verify in the same way the enclosure of the 9 zones of $(M,\, N,\, N',\, M')$ into the 9 polygons stemmed from P. In order to finish the proof, we use the same method with $T_6 = (M,\, N,\, N',\, M')$, then with the trapezoids T_5, T_4, T_3, T_2, and T_1 defined at each step. The figure 4 shows the progress of the proof.

It is clear that starting with $E_1 \in P_0 = \{x + iy\,;\, 0.64 \leq x \leq 1.4,\, and\, -\frac{2}{5}x \leq y \leq \frac{2}{5}x\}$, i.e. the area into which $d_1 = 0$, the new algorithm converges (see figure 5).

One can notice that, in order to also prove the convergence of the original algorithm, we have defined a pentagon rather than a trapezoid at the iteration 2, and we have considered the special cutting out of the first iteration.

We have proved that starting with E_1 into P_0, the new algorithm converges. This proof require much less computations than the original one and it can be done by a pen-and-pencil method.

Actually, to define T_{k-1}, we compute the 9 antecedents of T_k (with $k = 7 \cdots 2$). The polygons considered having less than 5 vertices, we compute less than 9×5 transformations by $\tau_{d_k}^{-1}$ at each step. Thus the proof requires less



(a) step 5 (b) step 3

Figure 4. The steps 5 and 3 of the proof.

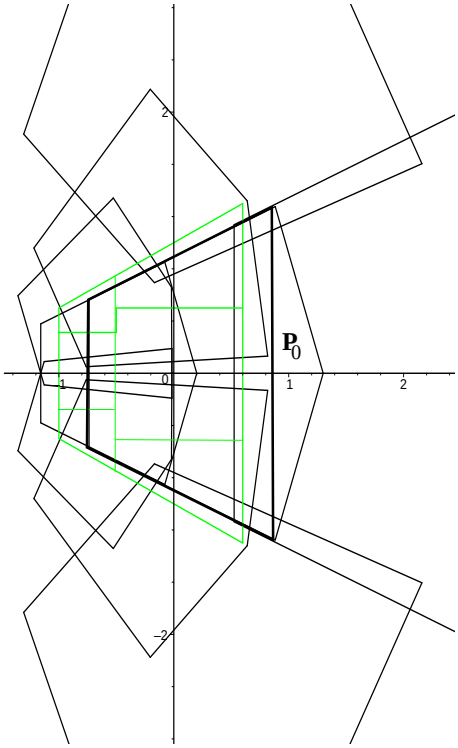


Figure 5. At step 1

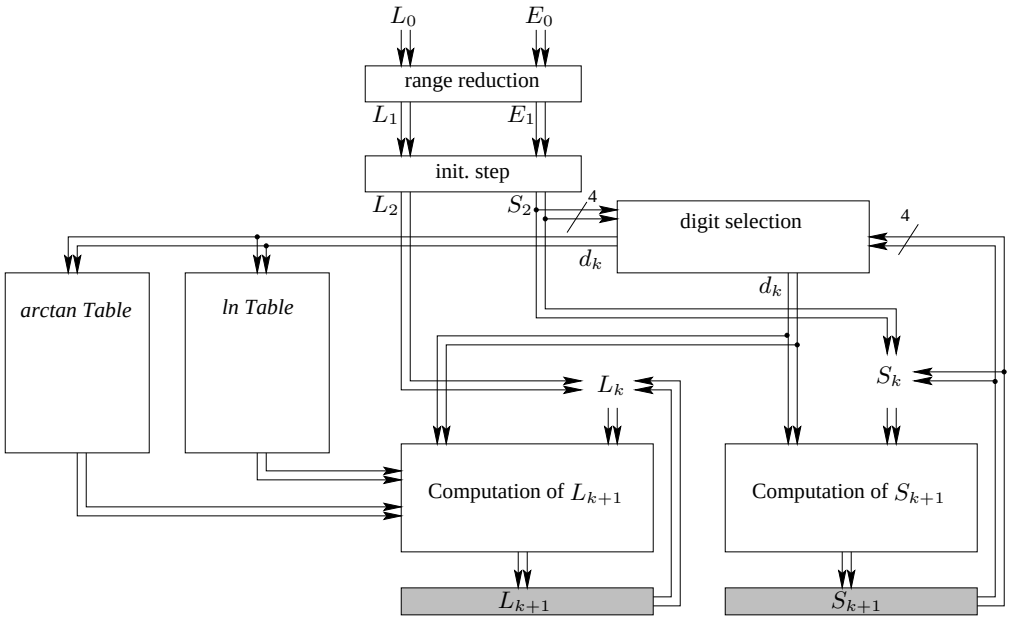


Figure 6. circuit for the L-mode of BKM

than $9 \times 5 \times 6 = 270$ computations of 2-D points. For symmetrical reasons we can approximatively divide by 2 the number of points to be computed.

5. HARDWARE IMPLEMENTATION

In this section, we present a brief description of the implementation for the calculation of $\ln(z)$. The architecture proposed here uses directly the algorithm presented above. As in previous sections, we consider only the L-mode of the algorithm. We shows in figure 5 the different components of a dedicated circuit for the L-mode.

The computations performed during a BKM iteration are:

- For the variable L:

$$L^r_{k+1} = L^r_k - \frac{1}{2}\ln\left[1 + d^r_k \times 2^{-k+1} + \left(d^r_k + d^i_k\right) \times 2^{-2k}\right] \tag{6}$$

$$L^i_{k+1} = L^i_k - d^i_k \arctan\left(\frac{2^{-k}}{1 + d^r_k \times 2^{-k}}\right) \tag{7}$$

- For the variable S:

$$S^r_{k+1} = 2\left(S^r_k + d^r_k\right) + 2^{-k+1}\left(S^r_k d^r_k - S^i_k d^i_k\right) \tag{8}$$

$$S^i_{k+1} = 2\left(S^i_k + d^i_k\right) + 2^{-k+1}\left(S^r_k d^i_k + S^i_k d^r_k\right) \tag{9}$$

We have shown that BKM allows the computation of the complex logarithm and exponential functions. Therefore, it makes it possible to compute many elementary functions of real and complex variables (complex exponential and logarithm functions, complex multiplication and division, complex functions $(ab)e^z$ and $\left(\frac{a}{b}\right)e^z$, real functions sin, cos, $\arctan\frac{y}{x}$, $\ln(x^2+y^2)$, $x\sqrt{a}$, $x\sqrt{a^2+b^2}$, $x/\sqrt{a^2+b^2}$, and 2-D rotations). You can refer to [4], for the details of the way these functions are obtained.

In order to obtain p significant bits, BKM roughly needs p iterations and the storage of $\frac{9p}{2}$ constants. BKM looks more complicated than CORDIC but the use of a redundant number system makes it more interesting since additions (subtractions) can be performed without carry propagation. Several authors suggested redundant versions of CORDIC, to get faster iterations [7, 8, 9, 10], but it requires a doubling of the iterations in space or in time.

6. CONCLUSION

We have presented a new version to the BKM algorithm which improves the original one, and which can be efficiently implemented on hardware. This algorithm becomes more interesting to implement than others redundant CORDIC. It allows to compute more functions without the drawback of the scaling factor. Elementary operations can be performed in redundant arithmetic. Thus we obtain an efficient implementation of the BKM algorithm. The drawback of the first iteration has disappeared without reduce the domain of convergence. Now at each step, the digit selection is made independently on the real and imaginary parts: i.e, the real (respectively imaginary) part of d_k only depends of the real (respectively imaginary) part of S_k. Furthermore, the range reduction is always made with only one shift-and-add operation. Unlike the original one which was very complicated, the new proof proposed in this paper is easy and can be shaped by hand.

REFERENCES

1. J. Volder, "The CORDIC computing technique," *IEEE Transactions on Computers* , 1959. Reprinted in E. E. Swartzlander, *Computer Arithmetic*, Vol. 1, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.
2. J. S. Walther, "A unified algorithm for elementary functions," *Joint Computer Conference Proceedings* , 1971. Reprinted in E. E. Swartzlander, *Computer Arithmetic*, Vol. 1, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.
3. J. M. Muller, *Elementary functions,* Algorithms and Implementation, Birkhäuser, 1997.
4. J. C. Bajard, S. Kla, and J. M. Muller, "BKM : A new complex algorithm for complex elementary functions," *IEEE Transactions on Computers* **43**, pp. 955–963, august 1994.
5. A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," *IRE Transaction on electonic computers* **10**, pp. 389–400, 1961.
6. J. M. Muller, "Discrete basis and computation of elementary functions," *IEEE Transactions on Computers* **34**(9), pp. 857–862, 1985.
7. H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundant implementation without correcting iterations," *IEEE Transactions on Computers* **45**, pp. 307–318, mar 1996.
8. M. D. Ercegovac and T. Lang, "Redundant and on-line CORDIC: Application to matrix triangularization and SVD," *IEEE Transactions on Computers* **39**, pp. 725–740, jun 1990.
9. N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Transactions on Computers* **40**, pp. 989–995, september 1991.
10. J. Duprat and J. M. Muller, "The CORDIC algorithm: New results for fast VLSI implementation," *IEEE Transactions on Computers* **42**, pp. 168–178, feb 1993.