1. Robert SMYK[1], 2. Maciej CZYŻAK[2]

Politechnika Gdańska (1), ANS w Elblągu (2)
ORCID: 1. 0000-0001-9365-4633

doi:10.15199/48.2025.01.09

New alpha max and beta min algorithm

***Abstract**. In this paper we present an improved version of alpha max plus beta min algorithm. We have proposed a new algorithm for approximating the alpha max and beta min coefficients. This leads to less complex algorithm formulas. Hardware architecture of alpha max plus beta min algorithm is also presented and analyzed.*

***Streszczenie.** W artykule przedstawiono ulepszoną wersję algorytmu alfa max plus beta min. Zaproponowaliśmy nowy algorytm aproksymacji współczynników alfa max i beta min. Prowadzi to do mniej skomplikowanych formuł algorytmów. Zaprezentowano i przeanalizowano architekturę sprzętową śtego algorytmu. (**Nowy algorytm: alfa max i beta min**)*

Keywords: *complex number magnitude, alpha max and betha min algorithm*.
Słowa kluczowe: *obliczanie modułu liczby zespolonej, algorytm alfa max and beta min*.

Introduction

The calculation of the magnitude of complex numbers is essential in processing signals from devices like Synthetic Aperture Radars (SAR) and spectrum analyzers. These devices typically use Fast Fourier Transform (FFT) processors, which output real and imaginary parts of harmonics. To obtain spectrum samples, the magnitudes of these harmonics must be calculated in real-time, necessitating high-speed circuits with the minimu delay. This demand limits the choice of algorithms and drives the continuous search for new or improved algorithms.

This work introduces a non-iterative approach for calculating magnitude with the controlled error, based on an improved version of the alpha max and beta min algorithm. The magnitude error is managed by selecting an appropriate number of approximation regions; more regions result in smaller errors. Previous literature detailed an approximation with two regions, but this work extends the algorithm to accommodate any number of regions. It establishes a relationship between maximum error and the number of regions, enabling the determination of the necessary regions for a specified error. The key to the improved algorithm is selecting the correct approximation region, guided by the quotient of arguments.

In the first section we review magnitude calculation algorithms, in next section we present the generalized magnitude approximation algorithm and then we discuss experimental results and square rooter architecture.

Review of known algorithms

A magnitude calculation algorithm usually includes squaring of arguments and square rooting. Squaring is a relatively simple operation and can be performed by multiplication or for small-length binary words by look-up and addition. Therefore we review only these magnitude calculation algorithms that perform square-rooting in the form suitable for the fast hardware implementation.

Square rooting algorithms which are commonly used for hardware implementations can be divided into two distinct categories: iterative and non-iterative algorithms. The iterative algorithms are often used but they usually require a considerable number of iterations to obtain the square root with an acceptable accuracy. The non-iterative algorithms [1,2] are highly desirable but the maximum acceptable error bound is difficult to provide. The iterative algorithms can be divided into three main groups: Newton-Raphson [6,10] method based algorithms, digit-by-digit algorithms [3,4,10] and CORDIC algorithms [5,10].

In the Newton-Raphson method based algorithms an estimate of the square root is obtained in each iteration [9]. The main disadvantage of these algorithms is the dependence of the error upon the number of iterations and the starting point. These algorithms can be transformed into a non-iterative form but the general division and multiplication are needed that slow down the execution of the algorithm. Moreover, the calculation error for a fixed number of stages strongly depends upon the choice of the starting point.

In digit-by-digit algorithms one-digit result is produced in every step of calculations by inspecting the shifted partial remainder which is derived from previous digit selections. Two forms are known: the restoring algorithm and the non-restoring algorithm which does not restore the remainder and requires less hardware resources. In practice the modified versions of non-restoring binary shift-subtract algorithms are used [6-8].

The CORDIC based algorithms belong to the third class of iterative algorithms, which can be adopted for the efficient square rooting computation. The typical CORDIC implementation of the square rooter requires typically several dozen clock cycles to calculate the results, without including input and output data processing. The CORDIC hardware uses two adders and two shift registers at each computational stage. The number of stages depends on the type of arithmetic used and the dynamic range of the input signal. For small dynamic ranges, for example 11-12 bit, six to eight computational stages may be required to achieve the small error of the result. For greater dynamic ranges, for example 16-bit, the number of stages may increase to 11-12. After completing the basic computations the signal normalization is needed, that is a multiplication by a suitable constant. This constant is a product of factors emerging at every CORDIC step due to the nature of the algorithm.

If the radicand is the sum of squares, the alpha max plus beta min algorithm [1] can be applied. This algorithm in its original version uses up to two approximation regions and allows to compute the magnitude of a complex number without division and without iterations. In principle until recently there existed two variants of the algorithm [9] which allows to compute the magnitude with the error not exceeding 3.95%, and with 1% respectively. In the previous version alpha and beta formulas were derived assuming an approximation error criterion, which led to complicated form of the obtained equations. In this work, two versions of the algorithm for determining alpha and beta coefficients have been derived for various error criteria. In the first case the same dependence was used for the error function as in [9] but we obtained a simpler form of the formula for the coefficients. In the second case the different error criterion is used that leads to even more simplified formulas for the coefficients, but at the expense of a slight increase in the approximation error. Simple forms of mathematical relationships obtained in this way can be used when calculations are necessary on an ongoing form, e.g. the processing is performed using a microcontroller.

New alpha max plus beta min algorithm

The formula for calculating the magnitude of a complex number is $|z| = \sqrt{P^2 + Q^2}$ where $z = P + jQ$. Computation of $|z|$ requires squaring and taking the square root, which is inefficient *i.e.* in hardware.

Let us define

(1) $$x = \max(|P|, |Q|)$$
(2) $$y = \min(|P|, |Q|)$$

We can approximate the magnitude by

(3) $$\sqrt{x^2 + y^2} = \alpha\, x \ + \beta y$$

Substituting $\frac{x}{y} = tan\ \theta$ into (3) we get

(4) $$\sqrt{1\ + \tan^2\theta} = \alpha\ + \beta\ tan\ \theta$$

After rearrangement of terms we receive

(5) $$1 = \alpha\ cos\ \theta\ + \beta\ sin\ \theta$$

For each angle θ and a set of pairs (α, β) equation (5) should be satisfied. But we have to approximate a constant one in (5) for $\theta \in [0, \pi/4]$ by using one or several chosen pairs of (α, β). For more exact approximation there should be the distinct pair (α, β) for given angle θ. The assignment of the pair for each θ would be ineffective. For general case we shall adopt denotations: θ_s- start of approximation region, θ_e – end of the region. Therefore, the range $[0, \pi/4]$ is divided into a number of regions $(\theta_{s(i)}, \theta_{e(i)}), i = 1,2, \dots, n$ with the appropriate pair (α_i, β_i) assigned to each of them, that will give an approximation of (5) with an acceptable error. The approximation error for the *i*-th region is given by

(6) $$e_i(\theta, \alpha, \beta) = (\alpha - \alpha_i)\cos\theta + (\ \beta - \beta_i)\sin\theta,\ i = 1,2, \dots, n$$

where (α, β) are exact values that should be used for the given pair (x, y) and (α_i, β_i) are approximating values for the i-th region. Using (5) we can eliminate α, β in (6) what gives

(7) $$e_i(\theta) = 1 - \alpha_i \cos\theta + \beta_i \sin\theta,\ i = 1,2, \dots, n$$

Approximation of the magnitude has been simplified to approximating a constant value of one. The right-hand side of (7) can approximate this constant using various methods. In this work, we use two types of error approximation.

In the first version, we impose that the approximation error for both ends of the interval and at a certain θ_{max} within the interval is equal. This leads to the following equations

(8a) $$e(\theta_s) = e(\theta_{max})$$
(8b) $$e(\theta_e) = -e(\theta_{max})$$

where $\theta_{max} = 0.5(\theta_s + \theta_e)$ is chosen as the midpoint angle of the interval. Using (6) and (7) in (8a) and (8b), respectively, we receive two equations

(9a) $$1 - \alpha\cos\theta_s - \beta\sin\theta_s = 1 - \alpha\ cos\ \theta_{max} - \beta\ sin\ \theta_{max}$$
(9b) $$1 - \alpha\cos\theta_e - \beta\sin\theta_e = \alpha\ cos\ \theta_{max} + \beta\ sin\ \theta_{max} - 1$$

After transforming these equations, we obtain the first version of new algorithm for alpha and beta

(10a) $$\alpha = \frac{2(\sin\theta_{max} - \sin\theta_s)}{\sin(\theta_e - \theta_s) + \sin(\theta_{max} - \theta_s) - \sin(\theta_e - \theta_{max})}$$
(10b) $$\beta = \frac{2(\cos\theta_s - \cos\theta_{max})}{\sin(\theta_e - \theta_s) + \sin\ (\theta_{max} - \theta_s) - \sin\ (\theta_e - \theta_{max})}$$

In the second version we impose approximation error as

(11a) $$e(\theta_s) = e(\theta_e)$$
(11b) $$e(\theta_{max}) = 0$$

Similarly as before, using (6) and (7) in (11) we have

(12a) $$1 - \alpha\cos\theta_s - \beta\sin\theta_s = 1 - \alpha\ cos\ \theta_{max} - \beta\ sin\ \theta_{max}$$
(12b) $$\alpha\ cos\ \theta_{max} + \beta\ sin\ \theta_{max} - 1 = 0$$

Finally after transforming (12) we obtain new formulas for α, β

(13a) $$\alpha = \frac{\sin\theta_{max} - \sin\theta_s}{\sin\ (\theta_{max} - \theta_s)}$$
(13b) $$\beta = \frac{\cos\theta_s - \cos\theta_{max}}{\sin\ (\theta_{max} - \theta_s)}$$

The above equations for α and β have a simple form, which may be beneficial when these coefficients have to be calculated in real time.

Algorithm for calculating magnitude of complex number

In presented algorithm we always assume a constant number of regions. Certain values of $r = y/x$ are boundary points, where we switch the respective (α, β) using (10) or (13). Therefore, each approximation region starts at θ_s and ends at θ_e. The overall magnitude estimation procedure is specified in *Algorithm 1*. The number of approximation regions that directly affects the maximum approximation error should be specified at the beginning of the magnitude estimation algorithm.

Algorithm 1*. Computation of magnitude of complex number* $P + jQ$

1. Determine of $x = \max(|P|, |Q|)$, $y = \min(|P|, |Q|)$ for the pair (P, Q)
2. Calculate $r = y/x$
3. Determine the proper (α_i, β_i) for i-th approximation region, based on r
4. Compute magnitude value as $\alpha_i\ x\ + \beta_i y$

Example 1. We are going to calculate $|z| = \sqrt{P^2 + Q^2}$ for $P = 2040$ and $Q = 1340$ which, when calculated directly, gives the result $|z| = \sqrt{2040^2 + 1340^2} = 2440.7376$. We will use our approximation algorithm using four intervals. For $\mathrm{r} = 1340/2040 = 0.65$, we have to apply third group of coefficients using (10), so $|z| = 0.9095\ \cdot\ 2040 + 0.4301 \cdot 1340 = 2431.714$. This gives us the relative error at the level $(2431.71 - 2440.73)/2440.73 = -0.00369$.

Experimental results

Using (10) and (13) the pairs of (α, β) coefficients can be pre-calculated as shown in Table 1 and Table 2. When implementing the algorithm, these coefficients can be stored in an array.

Table 1. Calculated values of (α_i, β_i), $i = 2,4,8$ based on (10)

Num. of regions	Angle range $\theta = \arctan(Q/P)$	α, β coefficients
2	[0, π/8)	α=1.0196 β= 0.1004
	[π/8, π/4)	α= 0.9035, β= 0.483
4	[0,π/16)	α= 1.0048, β= 0.0494
	[π/16,π/8)	α= 0.9759, β= 0.2445
	[π/8,3π/16)	α= 0.9095, β= 0.4301
	[3π/16,π/4]	α= 0.8081, β= 0.5993
8	[0,π/32)	α= 1.0012, β= 0.0246
	[π/32,π/16)	α= 0.994, β= 0.1226
	[π/16,3π/32)	α= 0.9772, β= 0.2194
	[3π/32,π/8)	α= 0.951, β= 0.3142
	[π/8,5π/32)	α= 0.9156, β= 0.4059
	[5π/32,3π/16)	α= 0.8714, β= 0.4936
	[3π/16,7π/32)	α= 0.8188, β= 0.5767
	[7π/32,π/4]	α= 0.7584, β= 0.6542

Table 2. Calculated values of (α_i, β_i), $i = 2,4,8$ based on (13)

Num. of regions	Angle range $\theta = \arctan(Q/P)$	α, β coefficients
2	[0, π/8)	α= 1.0, β= 0.0985
	[π/8, π/4)	α= 0.8862, β= 0.4737
4	[0,π/16)	α= 1.0, β= 0.0491
	[π/16,π/8)	α= 0.9712, β= 0.2433
	[π/8,3π/16)	α= 0.9051, β= 0.4281
	[3π/16,π/4]	α= 0.8042, β= 0.5964
8	[0,π/32)	α= 1.0, β= 0.0245
	[π/32,π/16)	α= 0.9928, β= 0.1224
	[π/16,3π/32)	α= 0.976, β= 0.2192
	[3π/32,π/8)	α= 0.9498, β= 0.3138
	[π/8,5π/32)	α= 0.9145, β= 0.4054
	[5π/32,3π/16)	α= 0.8703, β= 0.493
	[3π/16,7π/32)	α= 0.8178, β= 0.576
	[7π/32,π/4]	α= 0.7574, β= 0.6534

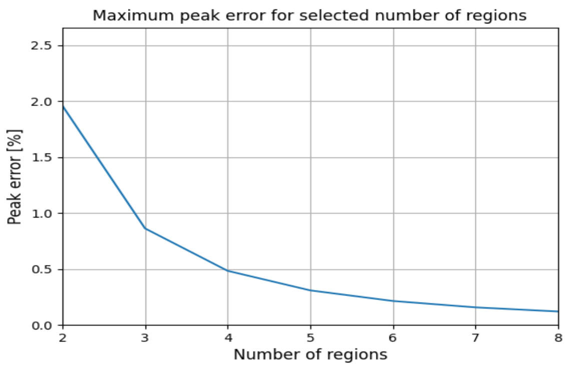

Fig. 1. Error for first version of α, β algorithm (10)

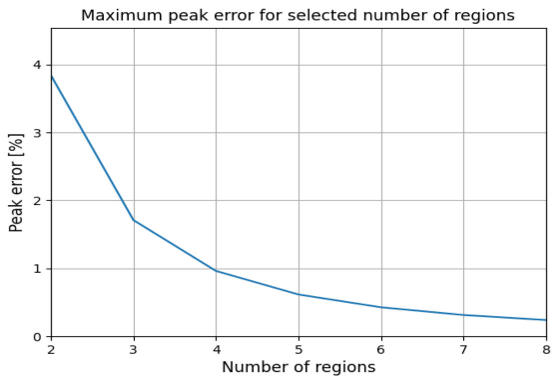

Fig. 2. Error for second version of α, β algorithm (13)

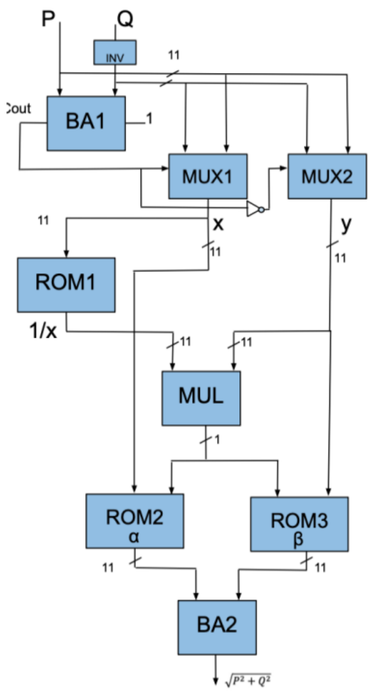

Fig. 3. Square rooter architecture diagram

In Fig. 1 and Fig. 2 the relative errors for selected number of regions for the first (10) and second (13) versions of algorithm are depicted. Using (13) the relative error is twice of that from (10). This is due to the approximation criterion, using (10) the computation formulas are much more complex, than in (13). When the simplicity is important (13) should be used and for better accuracy (10).

Square rooter architecture

In Fig. 3 square rooter architecture for 11-bit arguments is depicted. BA1 adder performs $P - Q$ subtraction using two's complement. Its output carry controls MUX1 and MUX2 multiplexers. If $P < Q$ then $C_{out} = 1$ and $x = Q$, $y = P$, else if $P > Q$ then $C_{out} = 0$ and $x = P$, $y = Q$. Then using ROM1 $1/x$ is computed. In the next step the obtained inverse of x is multiplied by y to obtain the number of region. Once the region is known ROM2 and ROM3 multiply x and y by α, β respectively. Finally addition is performed by BA2 as in (3).

Summary

This paper is an extension of a previously presented work [9]. The new formulas for α, β coefficients are shown for the error criterion used in [9]. Moreover, the new algorithm for computing α, β using the new error criterion is shown that gives considerably simplified formulas for α, β. In the first approach the formulas for α, β are more complex, but the error is smaller than in the second approach. The complexity of the formula may be important when the formula has to be computed in real-time. Moreover, a simple architecture of the square rooter is presented.

Authors: *dr inż. Robert Smyk, Wydział Elektrotechniki i Automatyki, Politechnika Gdańska, ul. Narutowicza 11/12, Gdańsk, E-mail: robert.smyk@pg.edu.pl , dr hab. inż. Maciej Czyżak, Akademia Nauk Stosowanych w Elblągu, Instytut Informatyki Stosowanej im. K. Brzeskiego, ul. Wojska Polskiego 1, Elbląg, E-mail: m.czyzak@ans-elblag.pl;*

REFERENCES

[1] Filip A. Linear approximations to x2 + y2 having equiripple error characteristics. IEEE Transactions on Audio and Electroacoustics 1973, No. 21, 554–556.
[2] Czyżak M., Smyk R., FPGA realization of an improved alpha max plus beta min algorithm, Poznan University of Technology Academic Journals. Electrical Engineering No. 80 (2014): 151-160.
[3] Kosheleva O., Babylonian method of computing the square root: Justifications based on fuzzy techniques and on computational complexity. Fuzzy Information Processing Society, 2009.
[4] Ercegovac M., On Digit-by-Digit Methods for Computing Certain Functions. Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers, 2007. ACSSC, 338–342.
[5] Meher P., Valls J., Juang T.B., Sridharan K., Maharatna K., 50 Years of CORDIC: Algorithms, Architectures, and Applications. IEEE Transactions on Circuits and Systems I: Regular Papers 2009, No. 56, 1893–1907.
[6] Kabuo H., Taniguchi T., Miyoshi A., Yamashita H., Urano M., Edamatsu H., Kuninobu, S. Accurate rounding scheme for the Newton-Raphson method using redundant binary representation. IEEE Trans. on Computers 1994, No. 43, 43–51.
[7] Sutikno T., Jidin A., Idris N., Jidin A., A simple strategy to solve complicated square root problem in DTC for FPGA implementation. 2010 IEEE Symposium on Industrial Electronics Applications (ISIEA), 2010, 691–695.
[8] Sajid I., Ahmed M., Ziavras, S., Pipelined implementation of fixed-point square-root in FPGA using modified non-restoring algorithm. (2010) 2nd International Conference on Computer and Automation Engineering (ICCAE), 2010, No. 3, pp. 226–230
[9] Smyk, R., Czyżak, M., Improved magnitude estimation of complex numbers using alpha max and beta min algorithm (2016), Zeszyty Naukowe Wydziału Elektrotechniki I Automatyki Politechniki Gdańskiej, No. 51
[10] Parhami, B., Computer arithmetic, Oxford University Press, 2010.