POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 80 Electrical Engineering 2014

Maciej CZYŻAK*
Robert SMYK*

FPGA REALIZATION OF AN IMPROVED ALPHA MAX PLUS BETA MIN ALGORITHM

The improved version of the alpha max plus beta min square-rooting algorithm and its realization in the Field Programmable Gate Array (*FPGA*) are presented. The algorithm computes the square root to calculate the approximate magnitude of a complex sample. It is especially useful for pipelined calculations in the *DSP*. The improved version allows to reduce the peak error from about 4% to 0.33%. This is attained by determination of the approximate ratio of arguments and adequate selection of algorithm coefficients. Four approximation regions are used and hence four sets of coefficients. Also a Xilinx *FPGA* implementation for 12-bit sign magnitude numbers is shown.

KEYWORDS: square root computation, alpha max plus beta min algorithm, field-programmable gate array

1. INTRODUCTION

In many digital signal processing applications the computation of the square root is a common task. In the general purpose computers the square root is usually computed using floating-point arithmetic library function or an arithmetic coprocessor. The IEEE floating-point standard assigns square-rooting to the group of basic arithmetic operations along with the usual four. Like division the square rooting can be can be formulated as a sequence of shift and subtract operations. This leads to the well-known non-restoring and restoring binary shift/subtract algorithms [1]. The other approaches may use the Newton-Raphson iteration or the CORDIC algorithm [1]. The square-rooting problem was considered in several works[2 - 9]. The common feature of these algorithms is the need to perform certain number of steps or iterations but calculation of the square root in digital signal processing may impose specific requirements. First the fixed-point arithmetic is commonly used and computations are performed using hardware, secondly the dynamic range of the signal usually does not exceed 16 bits. The main parameter of the square root algorithm along to the execution time or attainable pipelining rate is the peak absolute error for signals from the given dynamic range. The need to perform iterations increases the size of the square-rooter and may introduce the considerable delay. If radicand is a sum two-squares, the alpha max plus beta min algorithm can be applied.

* Gdansk University of Technology.

2. BASIC ALPHA MAX PLUS BETA MIN ALGORITHM

The basic form of this algorithm was presented by Filip [10]. The problem is to compute $R = \sqrt{P^2 + Q^2}$ where P and Q represent the quadrature pair, *i.e.*, the real and imaginary part of a complex number. Define

$$x = \max\left(|P|, |Q|\right) \tag{1}$$

$$y = \min\left(|P|, |Q|\right) \tag{2}$$

This transformation rotates the complex number $z = P + jQ$ in such way that the number obtained after rotation has the argument θ limited to the interval $[0, \pi/4]$. Thus

$$R = \sqrt{x^2 + y^2}, \; 0 \le \theta \le \pi/4 \tag{3}$$

We want to approximate R as

$$\hat{R} = ax + by, \tag{4}$$

It can be shown that

$$\hat{R} = a\cos\theta + b\sin\theta. \tag{5}$$

If for certain x, y and a, b we have $\hat{R} = R$ then

$$\sqrt{x^2 + y^2} = ax + by \tag{6}$$

Regarding that $tan\ \theta = y/x$, we obtain

$$1 = a\cos\theta + b\sin\theta, \tag{7}$$

That means, in general, that we have to approximate the constant value equal to 1 by a sum of two products. The approximation error is equal to

$$e(\theta) = 1 - a\cos(\theta) + b\sin(\theta) \tag{8}$$

This approximation is shown in Fig. 1.

In order to obtain the equiripple approximation we have to equalize the approximation error. We have

$$|e(\theta_0)| = |e(\theta_{max})| = |e(0)|. \tag{9}$$

Assuming that

$$\theta_{\max} = \frac{\theta_0}{2} \tag{10}$$

we obtain for $(0 \le \theta \le \theta_0)$

$$a = \frac{2}{1 + \sec\left(\theta_0/2\right)} \tag{11}$$

$$b = 2\tan\left(\theta_0/4\right) \tag{12}$$

$$|e_{\max}| = \tan^2\left(\theta_0/4\right) \tag{13}$$

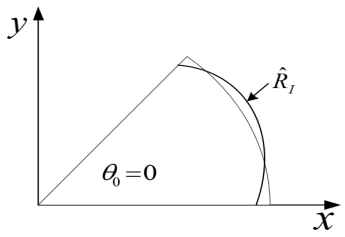

Fig. 1. Approximation using one region

For $\theta_0 = \pi/4$, we receive $a = \dfrac{2}{1+\sec(\pi/8)} = 0.9604$, $b = 2\tan(\pi/16) = 0.3978$ and $|e_{\max}| = \tan^2(\pi/16) = 0.039566$. We see that the error is about $\pm 3.95\%$ that, for example, for 12-bit signed numbers gives the maximum error from the interval [-80.8, 80.8]. In order to reduce this error, two regions of approximation can be introduced.

If we divide the range of approximation into two regions : Region I ($0 \le \theta < \theta_0$) and Region II ($\theta_0 \le \theta \le \pi/4$), then

$$R_{\mathrm{I}} = a_1 x + b_1 y \tag{14}$$

$$R_{\mathrm{II}} = a_2 x + b_2 y \tag{15}$$

(a_1 and b_1 for the Region I can be obtained using (11)-(12)) with $\theta_0 = \pi/8$.

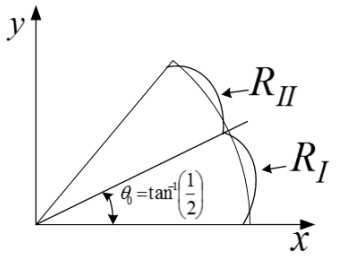

Fig. 2. Approximation using two regions

We obtain for Region II

$$a = \frac{\sqrt{2}\left(1-\sqrt{2}\sin\theta_0\right)}{2\sin\frac{1}{2}\left(\frac{\pi}{4}-\theta_0\right)+\cos\left(\frac{\pi}{4}+\theta_0\right)} \tag{16}$$

$$b = \frac{\sqrt{2}\left(\sqrt{2}\cos\theta_0-1\right)}{2\sin\frac{1}{2}\left(\frac{\pi}{4}-\theta_0\right)+\cos\left(\frac{\pi}{4}+\theta_0\right)} \tag{17}$$

$$|e_{\max}| = \tan^2 \frac{1}{4}\left(\frac{\pi}{4} - \theta_0\right) \tag{18}$$

$$\theta_{\max} = \frac{1}{2}\left(\theta_0 + \frac{\pi}{4}\right) \tag{19}$$

Using (11)-(13) for $\theta_0 = \pi/8$ and $\theta_{\max} = \pi/16$, we get

$$a_1 = 0.9902\ ,\ b_1 = 0.196983,\ |e_{max1}| = tan^2(\pi/32) = 0.0097\ .$$

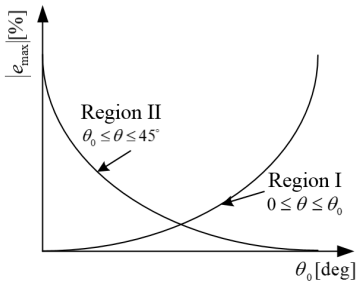

Fig. 3. Approximation error for the case of two approximation regions

We see that the error is about $\pm 0.9701\%$ that, for example, for 12-bit signed numbers gives the maximum error belonging to [-19.85, 19.85].

Using (16) and (17) we may calculate $a_2 = 0.839535$, $b_2 = 0.560960$ and

$$|e_{\max 2}| = \tan^2(0.25 \cdot (\pi/4 - \theta_0) = 0.0097\ .$$

3. GENERALIZED ALPHA MAX PLUS BETA MIN ALGORITHM

In Section 2 two versions of the algorithm have been presented. However, in certain situations the greater accuracy may be required and it can be attained by increasing the number of regions of approximation. For the general case we shall adopt the denotations:

θ_s - the start of the approximation region,

θ_e - the end of the region.

In order to obtain equal absolute values of the approximation error for both ends of the region and at a certain point θ_{max} within the region the following equations have to be fulfilled

$$|e(\theta_s)| = |e(\theta_{max})|\ , \tag{20}$$

$$|e(\theta_e)| = |e(\theta_{max})|\ . \tag{21}$$

It is advantageous to choose θ_{max}, as the midpoint of the region.

Using (8) we receive

$$1 - a \cdot \cos\theta_e - b \cdot \sin\theta_e = -1 + a \cdot \cos\theta_{\max} + b \cdot \sin\theta_{\max}\ , \tag{22}$$

$$1 - a \cdot \cos\theta_s - b \cdot \sin\theta_s = 1 - a \cdot \cos\theta_{\max} - b \cdot \sin\theta_{\max}\ . \tag{23}$$

After solving these equations we get

$$a = \frac{1}{2}\left(\cos\theta_e + \cos\theta_{\max} + \frac{(\cos\theta_s + \cos\theta_e)\cdot(\sin\theta_e + \sin\theta_{\max})}{\sin\theta_e - \sin\theta_s} \right), \quad (24)$$

and

$$b = a \cdot \frac{(\cos\theta_s - \cos\theta_e)}{\sin\theta_e - \sin\theta_s} \ . \quad (25)$$

The absolute error e_{max} can be expressed as

$$e_{\max} = |1 - a\cdot\cos\theta_e - b\cdot\sin\theta_e| = |1 - a\cdot\cos\theta_s - b\cdot\sin\theta_s| \quad (26)$$

As an example we shall use four approximation regions

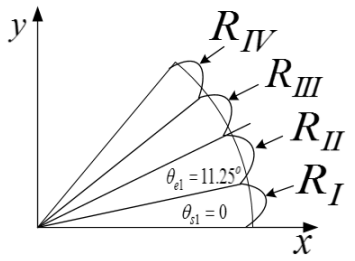

Fig. 4. Approximation using four approximation regions

For four approximation regions we obtain

$R_1 = [0, \pi/16)$, $R_2 = [\pi/16, \pi/8]$, $R_3 = [\pi/8, 3\pi/16]$, $R_4 = [3\pi/16, \pi/4]$.

For these regions we obtain the following coefficients

$a_1 = 0.997587$, $b_1 = 0.098254$
$a_2 = 0.959250$, $b_2 = 0.290985$
$a_3 = 0.884050$, $b_3 = 0.472534$
$a_4 = 0.774576$, $b_4 = 0.635924$

In Table1 the obtained approximation results are shown. We may observe that the maximum approximation error . We may observe that the maximum error has been reduced to 0.38% for $r = Min/Max = 0.43$ that means that the error has been reduced to one tenth of its value when only one region was used and to about one third for two regions.

Example 1. Compute $\sqrt{P^2 + Q^2}$ for P=2000 and Q = 1500.

We have $\sqrt{2000^2 + 1500^2} = 2500$. In this case $r = Q/P = 0.75$ and we have to apply the fourth group of the coefficients

$a_4 = 0.774576$, $b_4 = 0.635924$.

$$R = 0.774576\cdot 2000 + 0.635924\cdot 1500 = 1549.152 + 953.3886 = 2503.038$$

Hence the relative approximation error is equal to (2503.038-2500)/2500 = 0.00112, that is in good agreement with the respective error in Table 1.

Table 1. Square-root approximation using four regions

tan(P/Q)	Relation(A/Float)	Error[%]	tan(P/Q)	Relation(A/Float)	Error[%]
0.01	0.998520	+0.148038	0.51	1.002228	-0.222761
0.02	0.999352	+0.064777	0.52	1.002349	-0.234887
0.03	1.000085	-0.008468	0.53	1.002407	-0.240729
0.04	1.000717	-0.071691	0.54	1.002404	-0.240438
0.05	1.001249	-0.124892	0.55	1.002342	-0.234167
0.06	1.001681	-0.168083	0.56	1.002221	-0.222065
0.07	1.002013	-0.201285	0.57	1.002043	-0.204282
0.08	1.002245	-0.224525	0.58	1.001810	-0.180968
0.09	1.002378	-0.237842	0.59	1.001523	-0.152270
0.10	1.002413	-0.241280	0.60	1.001183	-0.118336
0.11	1.002349	-0.234896	0.61	1.000793	-0.079311
0.12	1.002188	-0.218752	0.62	1.000353	-0.035338
0.13	1.001929	-0.192919	0.63	0.999866	+0.013438
0.14	1.001575	-0.157476	0.64	0.999331	+0.066877
0.15	1.001125	-0.112509	0.65	0.998752	+0.124840
0.16	1.000581	-0.058114	0.66	0.997013	+0.298742
0.17	0.999944	+0.005610	0.67	0.997709	+0.229088
0.18	0.999214	+0.078552	0.68	0.998352	+0.164806
0.19	0.996706	+0.329390	0.69	0.998942	+0.105766
0.20	0.997689	+0.231113	0.70	0.999482	+0.051838
0.21	0.998576	+0.142420	0.71	0.999971	+0.002895
0.22	0.999368	+0.063224	0.72	1.000412	-0.041188
0.23	1.000066	-0.006568	0.73	1.000805	-0.080537
0.24	1.000671	-0.067055	0.74	1.001153	-0.115274
0.25	1.001183	-0.118342	0.75	1.001455	-0.145520
0.26	1.001605	-0.160542	0.76	1.001714	-0.171395
0.27	1.001938	-0.193770	0.77	1.001930	-0.193016
0.28	1.002181	-0.218150	0.78	1.002105	-0.210499
0.29	1.002338	-0.233809	0.79	1.002240	-0.223959
0.30	1.002409	-0.240879	0.80	1.002335	-0.233507
0.31	1.002395	-0.239497	0.81	1.002393	-0.239254
0.32	1.002298	-0.229803	0.82	1.002413	-0.241308
0.33	1.002119	-0.211942	0.83	1.002398	-0.239775
0.34	1.001861	-0.186062	0.84	1.002348	-0.234761
0.35	1.001523	-0.152315	0.85	1.002264	-0.226366
0.36	1.001109	-0.110853	0.86	1.002147	-0.214693
0.37	1.000618	-0.061835	0.87	1.001998	-0.199840
0.38	1.000054	-0.005419	0.88	1.001819	-0.181902
0.39	0.999418	+0.058234	0.89	1.001610	-0.160975
0.40	0.998710	+0.128962	0.90	1.001372	-0.137152
0.41	0.997934	+0.206600	0.91	1.001105	-0.110523
0.42	0.997090	+0.290982	0.92	1.000812	-0.081178
0.43	0.996181	+0.381943	0.93	1.000492	-0.049202
0.44	0.999492	+0.050806	0.94	1.000147	-0.014681
0.45	1.000095	-0.009547	0.95	0.999777	+0.022303
0.46	1.000626	-0.062559	0.96	0.999383	+0.061667
0.47	1.001084	-0.108380	0.97	0.998967	+0.103333
0.48	1.001472	-0.147160	0.98	0.998528	+0.147223
0.49	1.001790	-0.179049	0.99	0.998067	+0.193262
0.50	1.002042	-0.204199	1.00	0.997586	+0.241375

4. SQUARE ROOTER ARCHITECTURE AND FPGA REALIZATION

The hardware implementation of the algorithm comprises the determination of the greater and smaller value of the pair (P, Q). These values are termed *Max* and *Min*. Once they are known, their approximate quotient is computed in order to apply the proper pair of α and β coefficients. Subsequently these coefficients are multiplied by *Max* and *Min*, respectively and the sum of products is calculated. The general scheme is depicted in Fig. 5.

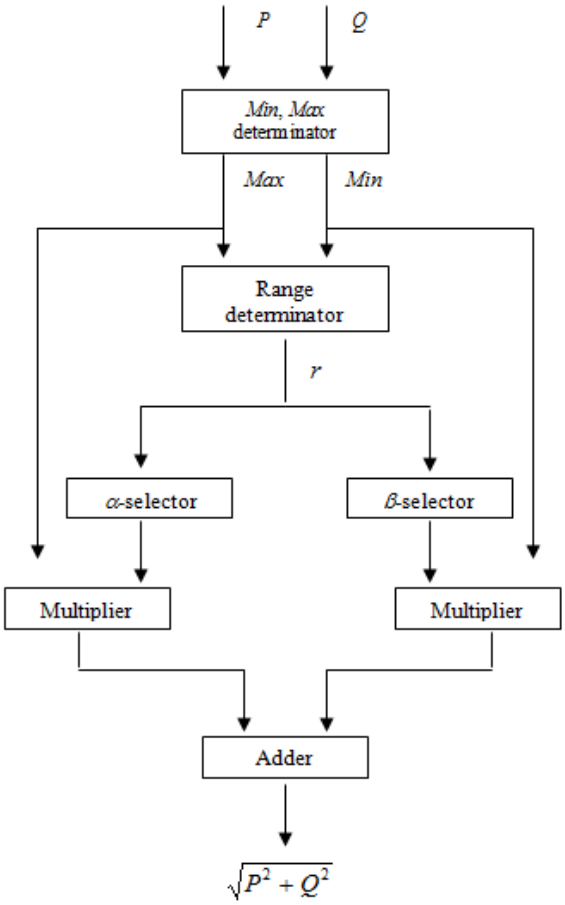

Fig. 5. The general scheme of the square rooter using four approximation regions

In Fig. 6 a detailed scheme of the square rooter implemented in the Xilinx *FPGA* is shown. In the first an binary adder is used that computes $P - Q$. If the result is nonnegative then $Max = P$ and $Min = Q$, for the negative result we have

the reverse order. The sign of the sum is used to control two multiplexers that select *Min* and *Max* values. The *ROM*1 computes the approximate reciprocal of *Max* using six most significant bits of its representation. Subsequently *MULT*1 computes the approximate quotient $r = \{Min/Max\}$.Only six msb's of *Min* are used. The obtained approximate value of r is used to detect one of four regions of approximation. The 6 msb's of the product are used to select by the look-up the proper pair of α and β. Once they are known the multiplications using *MULT*2 and *MULT*3 and final addition using the binary adder *BA*2 can be performed.

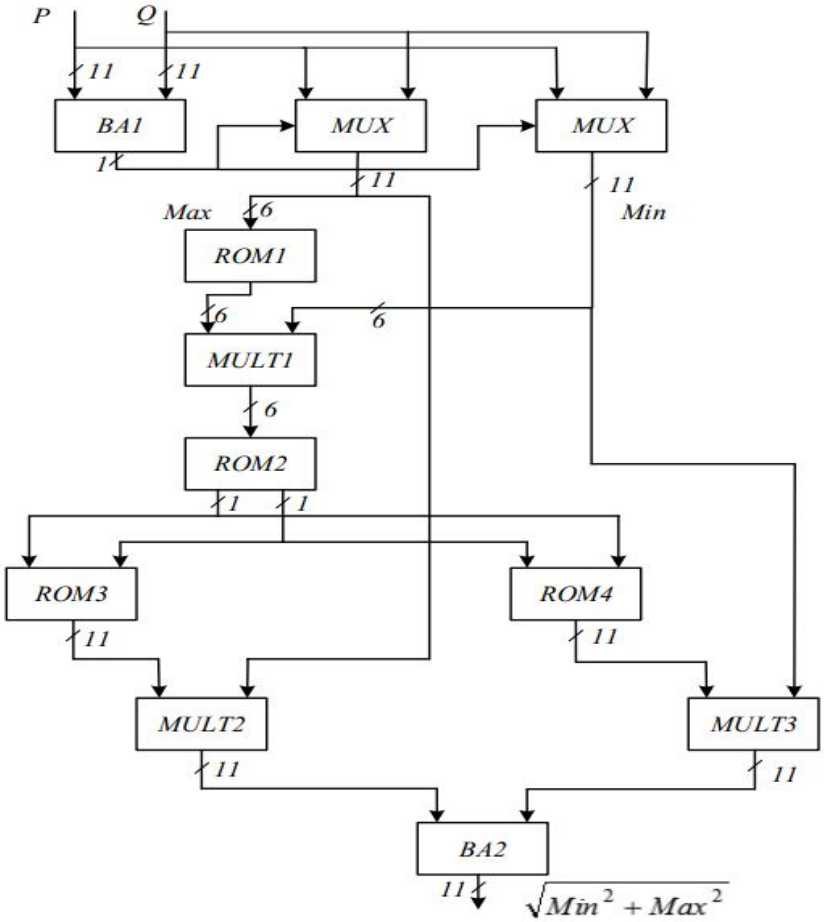

Fig. 6. The scheme of the *FPGA* square rooter that uses four approximation regions

The square rooter presented in Fig. 6 has been synthesized using Xilinx [11] design environment. Below we give the *FPGA* synthesis results.

Device utilization summary:

Selected Device : 6vcx75tff484-2
Slice Logic Utilization:
Number of Slice Registers: 33 out of 93120 0%
Number of Slice LUTs: 105 out of 46560 0%
Number used as Logic: 105 out of 46560 0%

Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 127
Number with an unused Flip Flop: 94 out of 127 74%
Number with an unused LUT: 22 out of 127 17%
Number of fully used LUT-FF pairs: 11 out of 127 8%
Number of unique control sets: 1

IO Utilization:
Number of IOs: 34
Number of bonded IOBs: 34 out of 240 14%

Specific Feature Utilization:
Number of BUFG/BUFGCTRLs: 1 out of 32 3%
Number of DSP48E1s: 2 out of 288 0%

Timing Summary:

Speed Grade: -2

Minimum period: 11.953ns (maximum frequency: 83.659MHz)
Minimum input arrival time before clock: 10.073ns
Maximum output required time after clock: 0.659ns

5. SUMMARY

The paper presents an improved version of the alpha max beta min algorithm that allows to compute the square of a complex number without division and without iterations. Up to date there existed two versions of the algorithm. The first one allowed to compute the square root with the error not exceeding 3.95%, whereas the second version permits to reduce the error to about 1%. The presented improved version attains about 0.33%. Also an Xilinx *FPGA* for 12-bit signed numbers implementation is shown.

REFERENCES

[1] Parhami B., *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, 2000.
[2] Zurawski J.H.P, Gosling J.B.: Design of a high-speed root, multiply and divide unit, IEEE Trans. on Computers, Volume.34, Number 1, pp. 13-23, 1985.
[3] Majerski S.: Square-root algorithms for high-speed digital circuits, IEEE Trans. on Computers, Volume 34, Number 8, pp. 1016-1024, 1985.
[4] Hashemian R.: Square rooting algorithms for integer and floating-point numbers. IEEE Trans. on Computers, Volume 39, Number 8, pp. 1025-1029, 990.
[5] Montushi P., Mezzalama M.: Survey of square-rooting algorithms. Proc. IEEE, *pt. E*, Volume 137, pp.31-40, 1990.
[6] Ciminera L., Montushi P.: High-radix square rooting, IEEE Trans. on Computers, Volume 39, Number 10, pp. 1220-1231, 1990.
[7] Ercegowac M.D., Lang T.: *Division and square-root: Digit recurrence algorithms and implementations*, Kluwer 1994.
[8] Sutikno T.: An efficient implementation of the nonrestoring square root algorithm in gate level, International Jourmnal of Computer Theory and Engineering, Volume 3, Number 1, pp. 46-51, 2011.
[9] Sutikno, Jidin A.Z., Jidin A., Idris N.R.N.: Simplified VHDL Coding of modified nonrestoring square root calculator, International Journal of Reconfigurable and Embedded Systems, Volume 1, Number.1, pp.37-42, 2012.
[10] Filip A.E.: Linear approximations to $\sqrt{x^2 + y^2}$ having equiripple characteristics. IEEE Trans. on Audio and Electroacoustics, Volume AU-21, Number 6, pp. 554-556, 1973.
[11] Xilinx, Virtex-7, www.xilinx.com.