XXVI Seminarium

ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE 2016

Oddział Gdański PTETiS

IMPLEMENTATION OF MAGNITUDE CALCULATION OF COMPLEX NUMBERS USING IMPROVED ALPHA MAX PLUS BETA MIN ALGORITHM

Robert SMYK[1], **Maciej CZYŻAK**[2]

1. Politechnika Gdańska, Wydział Elektrotechniki i Automatyki
   tel.: +48 58 347 13 32 e-mail: robert.smyk@pg.gda.pl
2. Politechnika Gdańska, Wydział Elektrotechniki i Automatyki
   tel.: +48 58 347 15 02 e-mail: maciej.czyzak@pg.gda.pl

Abstract: The paper presents the hardware implementation of the improved alpha max plus beta min algorithm for calculating the magnitude of complex numbers. This version of the algorithm requires the general division which is performed using a non-iterative multiplicative division algorithm. We analyze in detail the division algorithm, its error and the impact of finite word-length signal representations on the assumed total computation error. An analysis is performed to determine the binary length of operands at each stage of the magnitude calculator in order not to exceed the assumed total error. An FPGA implementation is presented along with its hardware requirement and delay.

Keywords: alpha max plus beta min algorithm, complex numbers magnitude, FPGA.

1. INTRODUCTION

Processing of digital complex signals may require the computation of magnitude and/or phase. In [1] we have presented a magnitude calculation algorithm being an extended and improved version of alpha max and beta min algorithm [2]. The new form allows to control accuracy of magnitude approximation by dividing the approximation interval into a certain number of subintervals called approximation regions. By increasing .the number of regions we may reduce the approximation error. Prior to implementation the number of regions has to be chosen which provides an acceptable maximum approximation error. During operation of the magnitude calculator the selection of the proper region is performed which determines the pair of coefficients used for magnitude calculation. For this purpose the quotient $r=y/x$ is used, where y is the imaginary part of the complex number and x the real part. This quotient r is computed using a non-iterative multiplicative division algorithm. In this algorithm an extended version of the reciprocal computation algorithm has been applied in order to limit the quotient approximation error. However, the hardware implementation requires the detailed analysis of all possible error sources. The total error resulting from using the algorithm from [1] has two components. The first one is the approximation error and the second is the incorrect region choice error. The first depends on the number of regions assumed for implementation. The second error results from the erroneous choice of the neighboring region instead of the correct one. The approximation error can be controlled only by increasing number of approximation regions, while the region choice error depends on accuracy of the estimate $\hat{r}$ of r for the given (x, y) pair. It can be remarked that the use of the standard division would be ineffective because of its computational complexity. We have applied the non-iterative division algorithm [4] that requires only four table look-up's, one addition and three multiplications. The division problem in this case is simplified because the dynamic range of both parts of the complex signal is usually limited to 12-bits in high-speed DSP systems. The primary application of the considered magnitude calculator can be at the output of the high speed FFT processor. However, the dynamic range in the presented implementation can be easily extended, for example, to 16 bits.

The magnitude approximation algorithm has been implemented using 12-bit fixed point arithmetic. The applied approach allows to compute the quotient with the maximum error not exceeding 0.3%. An analysis of the influence of the improper choice of the approximation region and its application for the determination of divider parameters is given. Additionally, we analyze the hardware implementation of the magnitude calculator in order to assess its amount of hardware and maximum attainable pipelining frequency. These parameters have been obtained from the synthesis for the FPGA using the VHDL description in the Xilinx environment [5].

The main contribution of the work is the synthesis of the circuit that computes the magnitude using the improved magnitude calculation algorithm. The synthesis encompasses the formulation and error analysis of the non-iterative division algorithm, magnitude error estimation and determination of the lengths of the binary arguments used within the algorithm. The proper lengths ensure that the maximum allowable error is not exceeded.

In Section 2 the principle of the magnitude calculation algorithm is reviewed. In Section 3 we present the computation of r error and its impact on the overall accuracy of the magnitude calculation. In Section 4 the non-iterative division algorithm for computation of r is presented. Section 5 contains the determination of reciprocal dynamic ranges for hardware realization that is shown in Section 6. Finally, results the synthesis results of magnitude calculator are presented in Section 7 and in Section 8 the discussion and comparison with other solutions is provided.

2. PRINCIPLE OF THE METHIOD

The improved magnitude calculation algorithm has been described in detail in the accompanying paper [1]. 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. The method calls for the computation of $x=\max(|P|,|Q|)$, $y=\min(|P|,|Q|)$ for (P,Q) pair, calculation of $r=y/x$, determination of the *i*-th proper approximation region, based on r, and the proper pair (α_i,β_i), $i=1,2,..,n$ and finally the computation of the magnitude as $\alpha_i x+\beta_i y$.

3. COMPUTATION OF *r* ERROR

The adequate approximation region is selected using an estimate $\hat{r}$ of $r=y/x$ for a given (x,y) pair. We have

$$\hat{r}=r+\varepsilon_r, \tag{1}$$

where ε_r is the division error. In the following, we will discuss the sources and impact of this error. We start by considering the magnitude approximation error $\varepsilon_{\max}$ in the vicinity of the boundary of two regions, which is the sum of the algorithm approximation error $e_{\max}$ and an error $\varepsilon_{\alpha,\beta}$ due to the use of the $(\alpha_{i+1},\beta_{i+1})$ pair of the neighboring region instead of the use (α_i,β_i) of the proper region. The total error $\varepsilon_{\max}$ can be expressed as

$$\varepsilon_{\max}=e_{\max}+\varepsilon_{\alpha,\beta}, \tag{2}$$

where $\varepsilon_{\alpha,\beta}$ for coefficients (α_i,β_i) and $(\alpha_{i+1},\beta_{i+1})$ of neighboring regions has the following form

$$\begin{aligned}\varepsilon_{\alpha,\beta}&=\alpha_i x+\beta_i y-(\alpha_{i+1}x+\beta_{i+1}y)\\&=x(\alpha_i-\alpha_{i+1})+y(\beta_i-\beta_{i+1})\end{aligned}. \tag{3}$$

Inserting $y=x\cdot r$ and $r=\hat{r}-\varepsilon_r$ into (3) we obtain

$$\begin{aligned}\varepsilon_{\alpha,\beta}&=x((\alpha_i-\alpha_{i+1})-r(\beta_i-\beta_{i+1}))\\&=x((\alpha_i-\alpha_{i+1})-(\hat{r}-\varepsilon_r)(\beta_i-\beta_{i+1}))\end{aligned}. \tag{4}$$

The general formula for n regions has the following form

$$\varepsilon_i^{\alpha,\beta}=\left|x(\alpha_i-\alpha_{i+1})-x\hat{r}(\beta_i-\beta_{i+1})+x\varepsilon_r(\beta_i-\beta_{i+1})\right|,\quad i=1,2,.3,..,n. \tag{5}$$

Moreover, at the common point at two adjacent approximation regions there is

$$\left|\alpha_i-\alpha_{i+1}+r_i(\beta_i-\beta_{i+1})\right|=0,\ i=1,2,3,...,n\ . \tag{6}$$

Using (5) and (6) we obtain

$$\varepsilon_i^{\alpha,\beta}=x\cdot(\beta_i-\beta_{i+1})\cdot\varepsilon_r,\ i=1,2,3,...,n\ . \tag{7}$$

$\varepsilon_i^{\alpha,\beta}$ attains its maximum for $x=x_{\max}$ and $x_{\max}=\max x_{i\in[1,n]}(\beta_i-\beta_{i-1})$.

We remind that $\varepsilon_{\max,i}^{\alpha,\beta}$ is a maximum error resulting from the use of α_{i+1} instead of the correct α_{i+1}. Hence for the given acceptable $\varepsilon_{\max,i}^{\alpha,\beta}$, we may determine the upper bound of ε_r which is given as

$$\varepsilon_r<\frac{\varepsilon_{\max,i}^{\alpha,\beta}}{x_{\max}\max(\beta_i-\beta_{i+1})},\ i\in[1,n]\ . \tag{8}$$

Remark that we impose the given upper bound for $\varepsilon_{\max,i}^{\alpha,\beta}$.

Example 1. Calculation of the lower bound of ε_r for $x_{\max}=2^{11}-1$ and $\max(\beta_i-\beta_{i-1})$, $i\in[1,4]$. For four regions β_i have the following values β_1= 0.0983, β_2= 0.2910, β_3= 0.4725, β_4= 0.6359 (Table 1 from [1]). Using (8) for the assumed $\varepsilon_{\max}^{\alpha,\beta}=0.24\%\cdot x_{\max}$ we receive

$$\varepsilon_r<\frac{4.91}{2047\cdot 0.19273}\cong 1.245\cdot 10^{-2}\ . \tag{9}$$

4. NONITERATIVE DIVISION ALGORITHM FOR COMPUTATION OF *r*

Computation of r requires general division which is implemented here by calculating first the approximate reciprocal $\hat{R}\cong 1/x$ with the subsequent multiplication $\hat{R}\cdot y$. The reciprocal is computed using the algorithm from [4]. This algorithm relies upon decomposition of the *m*-bit binary representation of $x\in[0,x_{\max}]$, where $x_{\max}=2^m-1$, into two shorter segments $\boldsymbol{a}$ and $\boldsymbol{b}$, with $a=x-|x|_{2^{k+1}}$ and $b=|x|_{2^l}$ with $m=k+l$. Such approach allows to use smaller look-up tables for the reciprocal calculation.

The principle of the division algorithm is given by the following formula

$$R=\frac{1}{x}=\frac{1}{a+b}=\frac{1}{a}-\frac{b}{a\cdot(a+b)}\ . \tag{10}$$

In order to compute the denominator of the right-hand term in (10) by the look-up table, *m*-bits would be required. But by replacing *b* by a certain constant K_1 only *k*-bits are needed

$$R\cong\hat{R}=\frac{1}{a}-\frac{b}{a\cdot(a+K_1)}\ . \tag{11}$$

Moreover, we assume that $K_1\in[0,b_{\max}]$, $b_{\max}=2^l-1$. K_1 is a suitably chosen constant that minimizes the maximum error of replacing $1/a(a+b)$ by $1/a(a+K_1)$. Now we shall evaluate the error ε_r resulting from this simplification

MOST WIEDZY Downloaded from mostwiedzy.pl

$$\varepsilon_r = R - \hat{R} = \frac{1}{a\cdot(a+K_1)} - \frac{b-K_1}{1+a/b}. \quad (12)$$

It is evident that ε_r attains its maximum with respect to a for $a = a_{\min}$ and with respect to b for $b = b_{\max}$

$$\varepsilon_r^{\max} = \frac{1}{a_{\min}\cdot(a_{m\min}+K_1)} - \frac{b_{\max}-K_1}{1+a_{\min}/b_{\max}}. \quad (13)$$

In order to estimate $\varepsilon_r^{\max}$, we have to multiply $\varepsilon_R^{\max}$ by the maximum dividend $y_{\max}$. We obtain

$$\varepsilon_r^{\max} = \varepsilon_R^{\max}\cdot y_{\max}. \quad (14)$$

First we shall review the simplified version of the method. This version does not give the sufficiently small reciprocal approximation error but it serves as an introduction to the final version.

The reciprocal approximation error resulting from using K_1 instead of b is expressed as

$$R - \hat{R} = \varepsilon(a,b,K_1) = \frac{b\cdot(b-K_1)}{a\cdot(a+b)(a+K_1)}. \quad (15)$$

As K_1 has to approximate b, it should belong to $[0,b_k]$, where $b_k = 2^k - 1$ is the end of the interval. It is evident that $\varepsilon(a,b,K_1)$ is maximum with respect to a when $a = a_{\min}$ and with respect to b when $b = b_k$ or for a certain $b = b_{\max}$. Using (15) and $a = a_{\min}$ the maximum division error for the maximum dividend $x_{\max}$ can be written as

$$\varepsilon_{div}^{\max} = \varepsilon(a_{\min},b,K_1)\cdot x_{\max} \quad (16)$$

The extreme of (26) with respect to $b_{\max}$ is obtained as

$$b_{\max} = -a + \sqrt{a(a+K_1)}. \quad (17)$$

Using $b_{\max}$, we want to equalize the division error, at the endpoint of the interval b_k and at a certain internal point $b_{\max}$

$$-\varepsilon(a_{\min},b_{\max},K_1) = \varepsilon(a_{\min},b_k,K_1). \quad (18)$$

The condition of equalization can be expressed as using (18)

$$-\frac{b_{\max}(b_{\max}-K_1)}{a_{\min}+b_{\max}} = \frac{b_k(b_k-K_1)}{a_{\min}+b_k}. \quad (19)$$

Inserting (17) into (19), we obtain the equation for K_1 that allows to determine such K_1 that provides the fulfillment of (17)

$$\left(1+\frac{4b_k}{a_{\min}}+\frac{4b_k^2}{a_{\min}^2}\right)K_1^2 + \left(\frac{2b_k^2}{a_{\min}} - \frac{4b_k^3}{a_{\min}^2} + 4b_k\right)K_1 - 4b_k^2 - \frac{4b_k^3}{a_{\min}} + \frac{b_k^4}{a_{\min}^2} = 0. \quad (20)$$

Example 2. Computation of $\varepsilon_r^{\max}$ for m=11, k=5 (binary length of a) and l=6 (binary length of b). Then for $a_{\min} = 64$, $b_{\max} = 63$ and $K_1 = 49.366$ we have

$$\varepsilon_R^{\max} = \frac{63-49.366}{1+64/63}\cdot\frac{1}{64(64+49.366)} = 9.3\cdot10^{-4} \ldots\ldots$$

In order to reduce the reciprocal approximation error, the term that approximates the error given by (12), can be used as the correction

$$\frac{1}{a+b} \approx \frac{1}{a} - \frac{1}{a\cdot(a+K_1)} - \frac{b(K_1-b)}{a(a+K_1)(a+K_2)} \quad (21)$$

To compute the third term in (21), by the table look-up with the use of a only, b has been replaced by a second constant K_2. After applying the correction as in (21), the reciprocal approximation error has the following form

$$\varepsilon(a,b,K_1,K_2) = \frac{b\cdot(K_1-b)\cdot(K_2-b)}{a\cdot(a+b)(a+K_1)(a+K_2)}. \quad (22)$$

Generally, we may receive an equiripple error ε_1 when the following conditions are fulfilled

$$\varepsilon_1(a_{\min},b_{\max 1},K_1,K_2) = \varepsilon_1(a_{\min},b_k,K_1,K_2), \quad (23a)$$
$$-\varepsilon_1(a_{\min},b_{\max 2},K_1,K_2) = \varepsilon_1(a_{\min},b_k,K_1,K_2), \quad (23b)$$

where $b_{\max 1}$ and $b_{\max 2}$ are the locations of the extremes of the error function. To solve these equations we have to determine $b_{\max 1}$ and $b_{\max 2}$ as functions of K_1 and K_2. However, we shall apply here an simplified form that will lead to the satisfactory solution by assuming that $K_2 = b_k$.

Computing the derivative of (22) we get

$$\frac{b_{\max 1}\cdot(K_1-b_{\max 1})\cdot(K_2-b_{\max 1})}{a_{\min}+b_{\max 1}} = \frac{b_{\max 2}\cdot(b_{\max 2}-K_1)\cdot(K_2-b_{\max 2})}{a_{\min}+b_{\max 2}}. \quad (24)$$

We shall determine the optimum K_1 using an iterative procedure by minimizing the difference between the left and right side of (24). Denoting the left side of (24) as $\varepsilon_L(K_1)$ and the right side as $\varepsilon_R(K_1)$ we shall try to find $K_1 \in [0,b_k]$ that minimizes $|\varepsilon_L(K_1) - \varepsilon_R(K_1)|$. This means that we have to find an approximate solution of the $|\varepsilon_L(K_1) - \varepsilon_R(K_1)| = 0$. This equation was numerically solved using the bisection method.

Next we shall calculate the values of extrema of ε_1. The approximate location of extrema is determined using the derivative of the nominator only. We have found that such way of determining of these extrema is sufficient with respect to the obtained maximum reciprocal error. The derivative of the nominator has the following form

$$3b^2 - 2(K_1+K_2)b + K_1K_2 = 0. \quad (25)$$

The solutions of (25) are equal to

MOST WIEDZY Downloaded from mostwiedzy.pl

$$b_{\max 1,\max 2} = \frac{(K_1 + K_2) \pm \sqrt{K_1^2 - K_1 K_2 + K_2^2}}{3}, \quad (26)$$

but we shall use the approximate solution using $K_2 = b_k$

$$b_{\max 1,\max 2} \cong \frac{(K_1 + b_k) \pm \sqrt{K_1^2 - K_1 K_2 + b_k^2}}{3}. \quad (27)$$

Using (27) in (24) we will find approximate solution of $|\varepsilon_L(K_1) - \varepsilon_R(K_1)| = 0$.

Example 3. The computation of ε_R for the modified reciprocal approximation method.

Regarding that ε_1 attains its extrema for $b_{\max 1}$, $b_{\max 2}$ for $a = a_{\min}$ and $K_1 = 27.959$, we obtain the following value of $\varepsilon_R^{\max}(b_{\max 1})$

$$\begin{aligned}\varepsilon_R^{\max}(b_{\max 1}) &= \frac{(b_{\max 1} - K_1)(b_{\max 1} - b_k)}{a_{\min}(a_{\min} + K_1)(a_{\min} + b_k)(1 + a_{\min}/b_{\max 2})} \\ &= \frac{(12.09 - 27.959)(12.09 - 63)}{64(64 + 27.959)(64 + 63)(1 + 64/12.09)} \\ &= 1.71741 \cdot 10^{-4}\end{aligned}, \quad (28)$$

and

$$\begin{aligned}\varepsilon_R^{\max}(b_{\max 2}) &= \frac{(b_{\max 2} - K_1)(b_{\max 2} - b_k)}{a_{\min}(a_{\min} + K_1)(a_{\min} + b_k)(1 + a_{\min}/b_{\max 2})} \\ &= \frac{(48.54 - 27.959)(48.54 - 63)}{64(64 + 27.959)(64 + 63)(1 + 64/48.54)} \\ &= 1.71731 \cdot 10^{-4}\end{aligned}. \quad (29)$$

The maximum of $\varepsilon_R^{\max}$ is attained for $x_{\max 1} = a_{\min} + b_{\max 1} = 64 + 49 = 76$ and for $x_{\max 2} = a_{\min} + b_{\max 2} = 64 + 49 = 113$. As it results from the principle of the alpha max and beta min algorithm the allowable dividend *y* is smaller than *x* at least by *1*. Thus we have $y_{\max 1} = 76 - 1 = 75$ and $y_{\max 2} = 113 - 1 = 112$. Hence we receive the maximum approximation error of $r = y/x$ as $\varepsilon_r^{\max} = \varepsilon_R^{\max} \cdot 112 = 0.019$.

Having $\varepsilon_r^{\max}$, we may determine the approximation error $\varepsilon_{\max}^{\alpha,\beta}$ resulting from the using the approximate division

$$\varepsilon_{\max}^{\alpha,\beta} = x_{\max} \cdot \max(\beta_i - \beta_{i+1})\varepsilon_r^{\max}, \; i \in [1,4]. \quad (30)$$

Since $x_{\max 1} = x_{\max 2}$ we receive

$$\varepsilon_{\max}^{\alpha,\beta} = 112 \cdot 0.192732 \cdot 0.019 = 0.41. \quad (31)$$

We can also find the maximum percentage error of the magnitude introduced by the approximate division

$$\varepsilon_{\max,perc}^{\alpha,\beta} = \frac{\varepsilon_{\max}^{\alpha,\beta}}{\sqrt{x_{\max 2}^2 + y_{\max 2}^2}} = \frac{0.41}{\sqrt{112^2 + 113^2}} 100\% = 0.25\%. \quad (32)$$

Concluding, we may remark that the used division algorithm does not substantially increase the magnitude approximation error (from 0.24% to 0.25%) resulting from the number of used approximation regions.

5. RECIPROCAL DYNAMIC RANGES FOR HARDWARE REALIZATION

In the fixed-point hardware implementation of the algorithm the limited binary length of reciprocal representation introduces an additional magnitude approximation error. We would like to determine the relationship between binary lengths and an error value.

In this section we shall determine the binary lengths of representations of terms on the right-hand side of (32). The allowable reciprocal errors $\varepsilon_R^{\max}(b_{\max 1})$ and $\varepsilon_R^{\max}(b_{\max 2})$ have been determined for the acceptable ε_r as in (19).

The reciprocal error in (21) can be expressed as

$$\varepsilon_R = \varepsilon_{R_1} + b\varepsilon_{R_2} + b(K_1 - b)\varepsilon_{R_3} \quad (33)$$

where ε_{R_i}, $i = 1,2,3$ denote errors of the nominators of the individual terms in (21). It is evident that ε_R attains its maximum for $b = b_{\max}$. Hence we have

$$\varepsilon_R = \varepsilon_{R_1} + b_{\max}\varepsilon_{R_2} + b_{\max}(K_1 - b_{\max})\varepsilon_{R_3}. \quad (34)$$

The lengths of binary representations of terms of (21) depend on the dynamic ranges and on ε_{R_i}, $i = 1,2,3$. ε_R must fulfill the inequality

$$\varepsilon_R < \varepsilon_{R\max}, \quad (35)$$

where $\varepsilon_{R\max}$ is a certain upper bound of the reciprocal error.

We may introduce certain error distribution for individual terms in (34). For example, we may assign a certain allowable error to terms in (34), for example, as: $\varepsilon_{R_1} < \varepsilon_{R\max}/2$, $\varepsilon_{R_2} < (\varepsilon_{R\max}/4) \cdot 1(/b_{\max})$ and $\varepsilon_{R_3} < (\varepsilon_{R\max}/4) \cdot (1/b_{\max}(K_1 - b_{\max}))$. Regarding the form of the binary representation of each individual term in (21) we have to consider its maximum and minimum values. The maximum value is determined by the smallest value of each of the denominators in (21) and the minimum value depends on the allowable error. The intervals of representations of individual terms can be determined as follows

1) for $1/a$ interval:

$$\left(2^{-\lfloor \log_2 a_{\min} \rfloor}, 2^{-\lceil \varepsilon_{R\max}/2 \rceil}\right) \quad (36)$$

2) for $b_{\max}/a(a + K_1)$ interval:

$$\left(2^{-\left\lfloor \log_2 \frac{a_{\min}(a_{\min} + K_1)}{b_{\max}} \right\rfloor}, 2^{-\left\lceil \frac{\varepsilon_{R\max}}{4 b_{\max}} \right\rceil}\right) \quad (37)$$

3) for $\dfrac{b_{\max}(K_1 + b_{\max})}{a_{\min}(a_{\min} + K_1)(a_{\min} + K_2)}$ interval:

MOST WIEDZY Downloaded from mostwiedzy.pl

$$\left(2^{-\left\lfloor \log_2 \frac{a_{\min}(a_{\min}+K_1)(a_{\min}+K_2)}{b_{\max}(K_1+b_{\max})} \right\rfloor}, 2^{-\left\lceil \frac{\varepsilon_{R\max}}{4b_{\max}|K_1-b_{\max}|} \right\rceil}\right) \quad (38)$$

Example 4. Calculation of the minimum and maximum ranges of exponents in (47) - (49).

Substituting $\varepsilon_{R\max} = 1.717 \cdot 10^{-4}$, $K_1 = 27.959$, $K_2 = 63$ and $a_{\min} = 63$ we have obtain:

for 1) we get $(2^{-6}...2^{-14})$,

for 2) we get $(2^{-6}...2^{-19})$,

for 3) we get $(2^{-10}...2^{-23})$.

6. FPGA HARDWARE REALIZATION OF MAGNITUDE CALCULATOR

In this section we shall present a hardware realization of the magnitude calculator in the Xilinx FPGA environment. The alpha max plus beta min algorithm with four approximation regions has been chosen. For the hardware implementation of the algorithm we have to consider an additional error introduced by the limited precision of algorithm coefficients, as we mentioned earlier. This error may cause that the improper selection of *r* will lead to the selection of (α, β) coefficients pair belonging to the neighboring interval instead of the right one. We remind that the maximum approximation error of the algorithm is 0.24%.

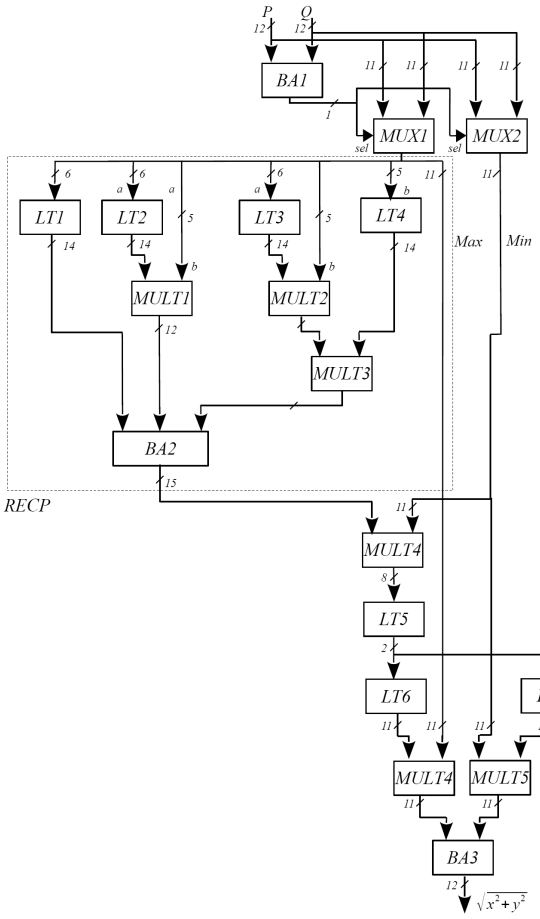

Fig. 1. Hardware implementation of magnitude calculator for four approximation regions

The first step of the hardware implementation of the algorithm (Fig. 1) includes the determination of $Max = \max(P,Q)$ and $Min = \min(P,Q)$. For this purpose subtraction of *P-Q* is performed using 12-bit two's complement binary adder (BA1). The sign of the sum is used to select *Min* and *Max* using two multiplexers MUX1 and MUX2. MUX1 selects *Max* and MUX2 selects *Min*. Once *Max* and *Min* are selected, the approximate reciprocal of *Max* is calculated within the RECP block and multiplied by *Min* using MULT4. The ranges of binary representation used in RECP block are given in Fig. 2. The product represents the approximate *r*. In the next step using *r* the adequate pair of (α, β) is selected (LT6, LT7) and this pair is used to calculate the approximate magnitude (MULT6, MULT7 and BA3).

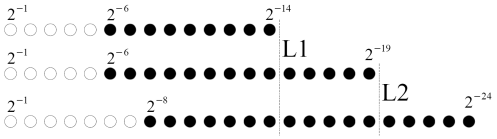

Fig. 2. Binary vectors used within the RECP

The crucial issue in hardware implementation is the computation of the approximate reciprocal of *Max* subsequently used for the computation of *r*. For this purpose the reciprocal determination algorithm from [4] was used. As shown in Section 5, the main goal of the algorithm is to decompose the binary representation of the divisor into two segments that can be used as the addresses of look-up tables utilized for computation of the reciprocal. In this way the look-up tables are smaller and faster. The *m*-bit binary representation of *x* is decomposed into one *k*-bit segment of *msb* bits and one *m-k*-bit segment of *lsb* bits. Such decomposition allows to compute the second term in (21) using only *k* bits and not *m* bits. If *m*=11 six-bit and five-bit look-up tables can be used in the RECP block. LT1 and LT2 tables store values of *1/a* and $1/a(a+K_1)$. The values of $1/a(a+K_1)(a+K_2)$ are stored in LT3 and for the $K_1 - b$ in LT4 and they are multiplied using the MULT3 multiplier. The required multiplication by *b* is performed using the MULT1 and MULT2. The product is subtracted by BA2 and finally, the value of *Min/Max* is obtained at the output of MULT4.

It should be mentioned that the dynamic ranges for the reciprocal analyzed before, have a direct impact on the stored words in LT1 - LT4 and binary lengths of MULT1, MULT2, MULT3. It can be seen that the vectors representing binary values of each factor are shifted to the right by 6bits, 8bits and 10bits, respectively. The subtraction algorithm realized by BA2 is illustrated in (Fig. 2). According to the analysis performed earlier, it results that the assumed accuracy can be achieved using 14-bit output words in the RECP block.

7. ANALYSIS OF HARDWARE COMPLEXITY AND TIME DELAY OF THE MAGNITUDE CALCULATOR

In Table 1 the synthesis results of implementation of the magnitude calculator in the Virtex 6 FPGA 6vcx75tff484-2 device are shown. The magnitude calculation requires three 4.48 ns clock cycles for three stage pipelining, and about 13.44 ns without pipelining.

MOST WIEDZY Downloaded from mostwiedzy.pl

Table 1. Hardware amount, delay and pipelining frequency of the magnitude calculator architecture for Xilinx FPGA

Device utilization and timing	Synthesis summary
Number of slice registers	224 out of 93120
Number of LUTs	118 out of 46560
Number of DSP48E1s	6 out of 288
Pipelining rate [ns]	4.48
Maximum pipelining frequency [MHz]	223.082
Delay [ns]	13.44

8. COMPARISON AND DISCUSSION

Before we compare the new method with other solutions some general remarks have to be made. The alpha max plus beta min algorithm in its original version has the maximum error equal to 3.98%, whereas the extended version of the algorithm allows to reduce the error almost freely. The original version requires only two multiplications by a constant and one addition. This is advantageous for both the ASIC and FPGA technology. For the smaller number ranges, eg. 12-bit number range, the magnitude calculator can be implemented using combinational circuits only. In ASICs it allows to attain very high pipelining frequencies limited only by the FA delay and power consumption. For example, for 40 nm CMOS technology this delay is about 75 ps [7], hence there are no architectural obstacles to attain gigahertz operating frequencies. The extended version of the algorithm calls for one additional division. In order to avoid the iterative realization, eg. using the digital recurrence or Newton-Raphson method [8], a non-iterative multiplicative division has been applied to reduce the delay and retain high pipelining rate. Now we shall compare the results from Table 1 with other solutions. In general, we may distinguish two approaches to magnitude calculation. With the direct approach the magnitude is computed for the given arguments as one stage process, while with the indirect approach initially the sum of squares of arguments is computed and next square rooting is performed. The algorithm presented in this work uses the former approach. A solution using the CORDIC algorithm was presented by the authors in [8]. The circuit uses five CORDIC iterations and requires 203 LUTs, the delay is equal to 10.035 ns that corresponds to the maximum pipelining frequency of about 100 MHz. The delay is smaller than that for the new method but at the cost of the maximum error equal to 2.48. This error could be reduced but at the cost of additional CORDIC iterations that would increase the delay.

For the purpose of comparison with indirect methods we shall review selected square-rooting algorithms only because the square-rooter and adder is the common part and we shall assume that it is implemented with three Xilinx DSP48E1. In dependence upon the Virtex 6 FPGA speed grade the delay belongs to the interval [1.66, 2.22] ns. Assuming speed grade -1 we have $t_{DSP48E1}$=2.22 ns. Therefore the total delay of the square rooter and adder is equal to t_{sqradd}=4.44 ns.

As the first example we shall shortly analyze square rooting using the Xilinx CORDIC IP core [6]. For magnitude calculation the real and imaginary parts have to be squared and added. It can be easily verified by numerical simulation of the CORDIC algorithm and 16-bit fixed-point arithmetic that to achieve the accuracy of 1ulp 12 iterations are needed. The Xilinx CORDIC core can operate with the maximum clock frequency of 339 MHz, which gives a 2.94 ns clock cycle. Assuming that one iteration requires 16-bit addition and 1-bit shift and is performed in one clock cycle, we obtain for 12 iterations the delay equal to

$$t_{sqradd} + 12 \cdot 2.94\,ns = 4.44ns + 35.28\,ns = 39.72ns\,.$$

The hardware amount of this core for square rooting is about 461 LUTs.

We shall also make a comparison with the CORDIC square-root realization in Altera Cyclone V that belongs to the similar FPGA class as the Xilinx Virtex 5. The operating frequency of NCO CORDIC IP Core is equal to 260 MHz and 838 ALMs [9] are needed. (ALM corresponds roughly to Xilinx LUT). The delay is equal to 2N+2 clock cycles where N is precision of magnitude in bits. For N=12 we receive 28 clock cycles, that gives the delay of 130 ns.

In [8] the results obtained for floating-point square rooting for numbers with 8-bit exponent and 23-bit mantissa in Virtex 4 4vfx100 have been given. The number of iterations was 25. The algorithm required 28 cycles for LogiCore with clock frequency 353 MHz and hardware amount of 464 slices (928 LUTs).

It can be concluded from the above that the new method has decisively smaller delay than iterative methods. Nevertheless, the hardware amount is comparable. Although the number of LUTs is much smaller but three DSP48E1 have been used. The important feature of the new circuit is also the feature, that the pipelining frequency can be increased by inserting additional pipeline register layers between the stages.

9. SUMMARY

The paper presents an implementation of the improved version of alpha max plus beta min algorithm for non-iterative calculation of the magnitude of complex numbers presented in [1]. This algorithm allows to control the magnitude approximation error but the total computation error in hardware is determined by the error resulting from the division algorithm and finite representations of internal signals used in the hardware implementation. We analyze in detail the error of division algorithm and the impact of finite word-length signal representations on the assumed total computation error. The error analysis is performed to determine such binary length of operands at each stage of the magnitude calculator, so that the assumed total error is not exceeded. We presented Xilinx FPGA implementation with the calculation delay about 14 ns. It is worthwhile to mention that the delay of the original Xilinx CORDIC IP core delay for magnitude calculation is about 35.28 ns.

10. REFERENCES

1. Smyk R., Czyżak M.: Improved magnitude estimation of complex numbers using alpha max plus beta min algorithm, this issue.
2. Czyzak, M.; Smyk, R.: FPGA realization of an improved alpha max plus beta min algorithm. Poznan University of Technology Academic Journals. Electrical Engineering 2014, 80, pp. 151-160.
3. Filip, A.: Linear approximations to $\sqrt{x^2 + y^2}$ having equiripple error characteristics. IEEE Transactions on Audio and Electroacoustics 1973, 21, pp. 554–556.

MOST WIEDZY Downloaded from mostwiedzy.pl

4. Czyzak, M.: Improved residue noniterative division for small numer ranges. International Scientific Conference on Computer Science and Engineering, Košice - Stará Lubovňa, Slovakia, September 20 - 22, pp. 178-185, 2010.
5. Xilinx Inc.: UG631 ISE Design Suite 14: Release Notes, Installation, and Licensing, October 2013.
6. Xilinx Inc. LogiCORE IP CORDIC v4.0. Product specification, 2011.
7. Zhang Y. et. al, Current-induced magnetic switching for high-performance computing in *Spintronics-based computing*, Eds. Zhao W., Prenat G., Springer, 2015.
8. De Dinechin F., Joldes M., Pasca B., Revy G., Multiplicative square root algorithms for FPGAs. 2010 International Conference on Field Programmable Logic and Applications (FPL), 31 Aug.-2 Sept.,2010, pp.574-577.
9. Altera, NCO IP Core User Guide, issued 2014.12.15.

IMPLEMENTACJA SPRZĘTOWA OBLICZANIA MODUŁU LICZBY ZESPOLONEJ Z WYKORZYSTANIEM ULEPSZONEGO ALGORYTMU ALPHA MAX PLUS BETA MIN

W artykule przedstawiono układową implementację ulepszonego algorytmu wyznaczania modułu liczby zespolonej. Wersja ta wymaga realizacji dzielenia sprzętowego. Zaproponowano wykorzystanie własnej nieiteracyjnej metody dzielenia. Wykonano szczegółową analizę algorytmu dzielenia pod kątem wyznaczenia wpływu skończonej długości reprezentacji binarnych sygnału wejściowego i sygnałów wewnętrznych układu na całkowity błąd dzielenia. Oszacowano również błąd całkowity obliczania modułu liczby zespolonej wynikający z wykorzystania nieiteracyjnej metody dzielenia. Ostatecznie wyprowadzono zależności pozwalające na dobór długości binarnej reprezentacji współczynników algorytmu dzielenia, przy której nie zostanie przekroczony maksymalny błąd obliczania modułu wynikający z właściwości numerycznych. Finalnie przedstawiono realizację rozwiązania układowego dedykowanego dla FPGA wraz z wynikiem syntezy w środowisku Xilinx.

Słowa kluczowe: algorytm alfa max beta min, moduł liczby zespolonej, FPGA.

MOST WIEDZY Downloaded from mostwiedzy.pl