Computing Pythagorean Triples in FPGA

Anatolij Sergiyenko, Anastasija Sergiyenko

National Technical University of Ukraine „KPI“, 37 Peremogy ave., Kyiv, Ukraine

aser@comsys.kpi.ua

Abstract. *A new method for calculating the Pythagorean triples is proposed, which is based on the two-step algorithm. The module, which implements this algorithm is configured in FPGA with small hardware volume, and can calculate the triple for a single clock cycle. Comparing to the CORDIC method this one has less hardware volume, higher speed, and provides the exact values of the sine and cosine results.*

Keywords

FPGA, CORDIC, twiddle factor.

1 Introduction

There are many methods for calculating of trigonometric functions in FPGA. One of the most used methods is based on the table function [1]. The interpolating polynome evaluation method needs a lot of additions and multiplications of the data with the large bitwidth [1,2]. Two decades the CORDIC method is widely used in FPGA to calculate the different trigonometric functions [1,3,4,12]. The only disadvantage of this method consists in that, that for n precise output data bits this algorithm takes n sequential iterations. As a result, the CORDIC algorithm has long latent delay of calculations.

In the presentation a new method for calculating the trigonometric functions in FPGA is proposed, which is based on the properties of the Pythagorean triples.

2 Pythagorean triples and ther properties

The solution of the Pythagorean problem consists in finding all square triangles with the natural sides a,b,c, i.e. in solving the Diophantine equation [5]:

$$a^2 + b^2 = c^2. \tag{1}$$

It represents the equation of the circle with the radius one, for which the following relation is satisfied:

$$\sin^2 \varphi + \cos^2 \varphi = 1, \tag{2}$$

where $\varphi = \mathrm{atan}(a/b)$. Equations (1),(2) show that for angles φ the trigonometric values are equal to the rational fractions:

$$\sin \varphi = a/c; \quad \cos \varphi = b/c; \quad \tan \varphi = a/b. \tag{3}$$

Therefore, one can get exact values of trigonometric functions (3) deriving the proper Pythagorean triple (a,b,c) for the given angle φ’. The problem is how to find the triple (a,b,c), which provides the satisfactory error ε of the angle representation. The Pythagorean problem can be solved by a set of methods [5,6]. All of them are combinatorial ones. Euclid has provided a formula for finding Pythagorean triples from any two positive integers m and n, $m > n$, namely:

$$a = m^2 - n^2; \quad b = 2mn; \quad c = m^2 + n^2. \tag{4}$$

Traversing m and n, we can find a couple m, n, which satisfy $\varepsilon \le |\varphi' - \varphi|$, where $\varphi = \mathrm{atan}(a/b)$. The searching area is minimized using the formula

$$\frac{n}{m} \approx \frac{\sqrt{r^2+1}-1}{r}, \tag{5}$$

where $r = \tan \varphi'$. Due to relations (4), (5) the calculation of trigonometric functions using the Pythagorean triple is more complex than the usual methods. But it is reasonable to form the table of triples, which are found for different angles. For the small angles $\varphi \approx \tan \varphi \approx 1/n$ the formulas are known, which depend on a single argument:

$$a = 2n + 1; \quad b = 2n^2 + 2n; \quad c = 2n^2 + 2n + 1; \tag{6}$$

$$a = 4n; \quad b = 4n^2 - 1; \quad c = 4n^2 + 1, \tag{7}$$

If we consider integer values of a,b then we get the generalized Pythagorean triples [7]. For such triples the operation of the angle addition is defined, which mimics complex multiplication. Consider two triples (a_1,b_1,c_1), (a_2,b_2,c_2) with the angles φ_1, φ_2, respectively. Then a new triple has the angle $\varphi = \varphi_1 + \varphi_2$:

$$(a,b,c) = (a_1,b_1,c_1) \oplus (a_2,b_2,c_2) = (a_1b_2 + b_1a_2, b_1b_2 - a_1a_2, c_1c_2). \tag{8}$$

The generalized Pythagorean triples with the angle addition operation $\oplus$ form the abelian group [5].

3 Using Pythagorean triples in computations

The equations (3) represent the exact values of the trigonometric functions. This means that these values can be used in computers without errors of data representation. For this feature the Pythagorean triples are used in some computer graphic tools, which provide the unexchanged image after a set of rotations [7]. Due to the fact that the precise sine and cosine values (3) are given by a triple of short integers, the tables of these functions have the minimized volume.

Some application specific processors, which are configured in FPGA, use the rational fraction data representation. The rational fraction *a/b* is the numerical object with integers in the numerator and the denominator. Rational fractions provide a simple set of arithmetical operations. The multiplication and division *a/b* to *c/d* are equal to *ac/*(*bd*), and *bd/*(*ac*), respectively. Addition of them is equal to (*ad+bc*)/(*bd*). The rational fractions stay in between integers and floating point numbers due to the precision of the data representation [8].

A set of processors for solving the linear algebra problems was designed, which showed the high effectiveness of the rational fraction data representation [9,10]. The discrete Fourier transform (DFT) is based on the multiplication to the twiddle factors. When these factors are based on the Pythagorean triangles, then the DFT error is minimized dramatically. The processor of the recursive DFT, which is based on this principle, is described in [11]. It is proven, that such an implementation of recursive DFT is much effective than the processors based on the well-known Goercel algorithm.

The disadvantage of the trigonometric function representation by the equations (3) consists in the complex procedure of finding the proper Pythagorean triangle for the given angle φ'. Below a new method is shown, which simplifies this triangle searching, and is implemented in FPGA.

4 Calculating Pythagorean triples in FPGA

The algorithm, which is implemented in FPGA, could not depend on the input data. If the algorithm has a set of stages, then the number of them has to be predefined. Therefore, the algorithms, based on the combinatorial search, usually do not fit the FPGA implementation. Among them the most of Pythagorean triple searching algorithms are.

The two-staged schema of the Pythagorean triangle search is proposed. It is based on the operation of angle addition (8). The given angle φ' is represented by the sum of angles $\varphi' = \varphi'_1 + \varphi'_2$. At the first stage the triple (a_1,b_1,c_1) is searched, for which the angle $\varphi_1 = \varphi'_1 + \delta_{\varphi1}$ differs from the given value φ'_1 to the difference $\delta_{\varphi1}$. At the second stage the triple (a_2,b_2,c_2) is searched for the angle $\varphi_2 = \varphi'_2 - \delta_{\varphi1}$. The resulting Pythagorean triangle is calculated by the formula (8). It represents the angle $\varphi = \varphi_1 + \varphi_2 = \varphi'_1 + \varphi'_2 + \delta_{\varphi2}$, where $\delta_{\varphi2}$ is the error of the angle φ' representation.

The angle φ'_1 is represented by higher bits of the code φ', and the angle φ'_2 is represented by lower bits of it. Then the first stage of the schema can be implemented in ROM, for which the code φ'_1 is an address input. Consider that the least significant bit of the code φ'_2 is equal to $\pi/2^{15}$, i.e. $\delta_{\varphi2} < \pi/2^{16}$, and we take into account, that $n \approx 1/\tan\varphi_2$. Then the second stage can be calculated by the formula (6), when $\varphi_2 < \pi/512$, and by the formula (7), when $\varphi_2 < \pi/1024$.

The structure of the module, which searches for the Pythagorean triple for the angles $0 < \varphi' < \pi/4$, is shown in the fig.1. The input phase code is stored in the register RGP. The most significant bits of the code select a Pythagorean triple (a_1,b_1,c_1), and the angle error value $\delta_{\varphi1}$ in ROM1. The corrected value φ_2 of the angle φ'_2, which is given by the least significant bits of RGP, is formed by the adder SM1, and selects the second Pythagorean triple (a_2,b_2,c_2) in ROM2. The multipliers MPU, and adders SM1, SM2 calculate the equation (8).

For example, consider $\varphi' = \pi/6$. It is represented by the code $85 \cdot 2^7 + 43$ in RGP. The triple (120,209,241), and the error code -7 are read from ROM1 by the address 85. The code to select the second triple is $\varphi_2 = \varphi'_2 - \delta_{\varphi1} = 43+7 = 50$. This code represents the angle 0.0024. Then the factor n is equal to $n =]1/\tan(0.0024)[= 417$. Due to the formulas (6), the second Pythagorean triple is equal to (835, 348612, 348613), and it is read from the ROM2 by the address $\varphi_2 = 50$.

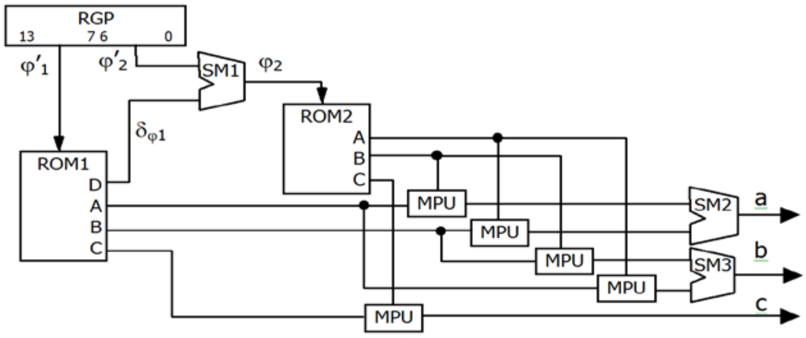

Fig. 1. Structure of the Pythagorean triple calculator

The resulting Pythagorean triple, calculated due to the formula (8), is equal to (42007955, 72759708, 84015733). It represents the angle $\pi/6$ with the error of $1.476{\cdot}10^{-5}$ radians, which is equal to 0.31 of the least significant bit in RGP.

The structure in the Fig.1 is implemented in the Xilinx Artix XC7A20SL, and occupies 235 LUT flip-flop pairs, and 8 multipliers. The maximum clock frequency achieves 135 MHz. This frequency can be much higher when the pipelining technique is used. If the fixed point results are needed then the division unit has to be attached, which hardware volume is four hundreds LUTs for the 16-bit result. The CORDIC module with the same precision has in four times higher hardware volume and the latent delay, which is higher in one degree of magnitude [12].

5 Conclusions

The Pythagorean triple provides the simple method of deriving the exact values of trigonometric functions. Such a triple can be found by a set of methods. But all of such known methods are combinatorial ones, and therefore, they are ineffective for implementing in FPGA. A new method for the calculating the Pythagorean triples is proposed, which is based on the two step algorithm. The module which implements this algorithm is configured in FPGA with small hardware volume, and can calculate the triple for a single clock cycle. This module can be used in the application specific processors for the linear algebra problem solving, digital signal processing, and in many other fields.

References

[1] B-G.Goldberg: Digital Frequency Synthesis Demystified. *LLH Technology Publishing*, USA: 1999.

[2] K.Hariharan, E. B. Hubert, K. V. O. Divyalakshmi, K. Shamalla, A. V. Kumar: Coherent Sinusoid Generation using Novel DDFS Architecture. *International Journal of Smart Home* 6(1):17-28, 2012.

[3] R. Andraka: A survey of Cordic algorithms for FPGA based computers. *ACM/SIGDA 6-th International Symposium on FPGAs*: 1981–2000, 1998.

[4] A. Sergyienko, O. Maslennikov: Implementation of Givens QR Decomposition in FPGA. *LNCS*. Berlin: Springer. V. 2328. 453–459, 2002.

[5] L. Tan: The Group of Rational Points on the Unit Circle. *Mathematics Magazine*, 69(3):163-171, 1996.

[6] Y. Thibault, Y. Kenmochi, A.Sugimoto: Computing Admissible Angles from Rotated Digital Images. *Combinatorial Image Analysis*, LNCS, Springer: 99-111, 2008.

[7] R.T. Farouki: Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable. *Series: Geometry and Computing*, Springer, 1: 2008.

[8] O. Maslennikow, P. Ratushniak, A. Sergiyenko: Generator opisow VHDL blokow operacyjnych dzialajacych w arytmetyce ulamkowej. *Prace XI Konferencji Naukowej „Reprogramowalne uklady cyfrowe”*, RUC’2008, Szczecin: 167-172, 2008.

[9] O. Maslennikow, N. Maslennikowa, P. Ratushnyak, A. Sergiyenko, M. Wozniak: Applicaton specific processors for the autorgressive signal analysis. *8-th Int. Conf. Parallel Processing and Applied Mathematic, PPAM'2009, Part I*, LNCS, Springer, 6068: 80–86, 2010.

[10] O. Maslennikow, V. Lepekha, A. Sergiyenko, A.Tomas, R.Wyrzykowski: Parallel Implementation of Cholesky LLT Algorithm in FPGA–Based Processor. *LNCS*, Springer, 4967: 137-147, 2008.

[11] A. Sergiyenko, V. Melkovska: A method of the signal demodulation with the DTMF modulation. *Bulletin of NTUU „KPI”: Informatics, Control and Computer Engineering*, 48: 82–84, 2008. (In Ukrainian)

[12] DS858, LogiCORE IP CORDIC v5.0, Product Specification. October 19, 2011, available at http://www.xilinx.com/