International Journal of Business and Management Invention
ISSN (Online): 2319 – 8028, ISSN (Print): 2319 – 801X
www.ijbmi.org || Volume 6 Issue 3 || March. 2017 || PP—50-54

Evaluation of CORDIC Algorithm for the processing of sine and cosine functions

Ajakida Eski[1], Denald Komici[2], Orion Zavalani[3]
[1](Ph.D candidate, Department of Electrotechnics, Politechnic University of Tirana, Albania)
[2](Msc, Department of Automation, Politechnic University of Tirana, Albania)
[3](Prof.Dr., Department of Automation, Politechnic University of Tirana, Albania)

ABSTRACT: *When performing real-time digital signal processing, the improvement in the speed of elementary-function operation and in the reducing of memory use becomes indispensable. In this paper, we have presented the choise of Coordinate Rotation Digital Computer (CORDIC) algorithm to realize the valuation of the sine and cosine functions compared to the polynomial approximation algorithm of Taylor series. Finally, a detailed comparison between the two methods show that CORDIC algorithm is 2.3 times faster and occupies 47.1% less memory . This means a smaller and cheaper design, a smaller power consumption.*
Keywords: *CORDIC, speed, trigonometric function,Vector Rotation*

I. INTRODUCTION

The real -time measurement performance is very dependent on digital signal processing. Although, the enormous improvements in computing technology have occurred, it is still important to reduce complicated operations to simple one. In digital signal processing, elementary function operations, such as trigonometric function are performed. How to make these elementary function operations more fast and efficient has become an important theory.

There are several algorithms that only use the four basic operations (+, −, ×, /) to find the sine, cosine, or tangent of a given angle. It is an important problem to get the value of sine and cosine functions of high precision , smaller area and faster in digital signal processing.

Calculating the values of the sine and cosine functions in polynomial approximation of Taylor series needs a lot of areas and many polynomials, hence the whole system will be complicated. The processing can be accelerated by using CORDIC algorithm, invented by Jack Volder in the late 1950s. The CORDIC method is a recursive algorithm that reduces the problem of computing apparently complicated functions, such as trigonometric functions, to a succession of simple operations. Specifically, these simple operations are shifting and adding. In this paper, processing of sine and cosine is fixed-point and in vectoring mode. The CORDIC arithmetic needs smaller area and can calculate faster the sine and cosine value of high precision. However, in spite of the age of the method, it is still important. The method is one of those great ideas that is able to survive despite technological changes due to the simplicity and efficient hardware implementation.

II. CORDIC ALGORITHM

CORDIC [1] is an iterative algorithm for computing trigonometric, hyperbolic and transcedental functions in a compute efficient manner. The CORDIC algorithm provides an iterative method of vector rotations by predefined angles using only shift and add operations. The conventional CORDIC is a high sequential algorithm in which the output values of the present iteration act as the input to the next iteration.
To explain the basic concept of CORDIC[2], consider a two –dimensional Euclidean space as shown in Fig.1.

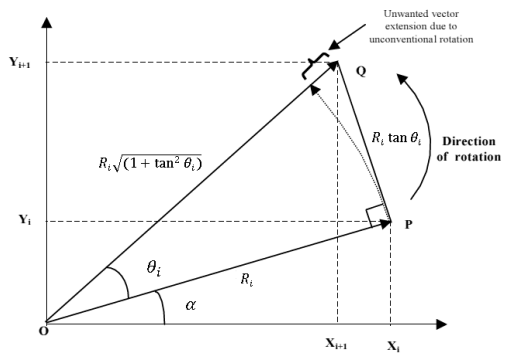

Fig 1. Typical computing step

Let X_i and Y_i be the x and y coordinates of the vector OP with magnitude R_i and an angle α. This vector is rotated through an angle θ_i to form the new vector OQ. The quantity i is equal to the number of particular step under consideration. The rotation is not a perfect vector rotation but a motion of a vector OP along the tangent of the circle formed by OP as radius at the point P. Then the resultant vector will have a magnitude given by $R_i\sqrt{(1+\tan^2\theta_i)}$. Either of two choises of direction produces the same change in magnitude therefore the increase in magnitude may be considered as a constant.

The CORDIC architecture express the coordinate X_{i+1}, Y_{i+1} of i^{th} micro-rotations as follows:

$$X_{i+1} = X_i - Y_i \tan\theta_i$$

$$Y_{i+1} = Y_i + X_i \tan\theta_i \quad (1)$$

In CORDIC algorithm the desired rotation angle θ is achived by a series of i micro-rotations (where $i = 0, \dots N$) by predefined angles wich are stored in a ROM. In simple words the input angle is decomposed into small micro-angles that take value

$$\theta_i = arctan(2^{-i}) \quad (2)$$

Taking into consideration the direction of rotations, $\pm$ notation may be represented by the binary s, where $s = +1$ for anticlockwise rotation and $s = -1$ for clockwise rotation. For an infinite number of iterative rotation angles, the relationship between θ_i and θ can be expressed as follows:

$$\theta = \sum_{i=0}^{\infty} s_i \cdot \theta_i = \sum_{i=0}^{\infty} s_i \cdot arctan\ (2^{-i})$$

Note that, by restricting the angular rotation magnitude in (2), X_{i+1} dhe Y_{i+1} may be obtained by two simultaneous shift and add operation, expressed as

$$X_{i+1} = X_i - s_i Y_i \cdot (2^{-i})$$

$$Y_{i+1} = Y_i + s_i X_i \cdot (2^{-i}) \quad (3)$$

The angle accumulator is defined as:

$$Z_{i+1} = Z_i - s_i \cdot arctan(2^{-i})$$

$$s_i = \pm 1$$

Since each iteration in CORDIC is not a perfect rotation, it modifies the length of the vector in a quantity of K_i expressed as

$$K_i = \sqrt{1 + s_i^2 2^{-2i}} \quad (4)$$

Therefore, after N iterations the vector is amplified in magnitude by a factor

$$K_N = \prod_{i=0}^{N-1}\left(\sqrt{1 + s_i^2 2^{-2i}}\right) \quad (5)$$

If $s_i \in [+1, -1]$, value of K remains same, irrespective of the polarity of rotation. Therefore, the value of K is a constant for a fixed number of iterations. In order to maintain a constant vector length and to obtain the correct values of X_N dhe Y_N, the factor K has to be compensated at the end of all iterations. The obtained result has to be scaled by $1/K_N$. However using consecutive rotations the scale factor can be pre-computed.

CORDIC algorithm consists of two operating modes [4], the rotation mode (RM) and the vectoring mode (VM). In the rotation mode a vector is rotated by an angle θ_i to obtain a new vector (X_{i+1}, Y_{i+1}). The direction of each subsequent micro-rotation is determined by the sign of the angle left for rotation after last micro-rotation. In every micro-rotation i, fixed angles of the value $arctan(2^{-i})$ which are stored in a ROM are subtracted or added from/to the accumulator angle Z_i , so that the accumulator angle approaches to zero . The vectoring mode accumulates the angles needed to rotate a given vector for minimizing Y_i. For this purpose, the vector is rotated towards the x-axis so that the y-component approaches to zero. The sum of all angle rotations is equal to the value of α, while the value of the x-component corresponds to the length R of the vector .

In the RM mode the direction of the micro-rotations s_i are determined by the sign of the Z_i variable, if sign of Z_i is positive $s_i = +1$ otherwise $s_i = -1$. In VM the decision criteria depends on the sign of the *Y* variable, if it is positive then $s_i = -1$ else $s_i = +1$.

Both methods initialize the angle accumulator with the desired angle. In the rotation mode , the angle accumulator is initialized with the desired angle denoted by Z_0 . The value of Z_0 is initialized to zero in vectoring mode.

The value selected for *N* is a function of the desired computing accuracy. To satisfy an *N*-bit precision CORDIC operation, *N*+1 iterations are nedded.

Note, that the CORDIC method as described performs rotations only within –π/2 and π/2 [ ]. This limitation comes from the use of 2^0 for the tangent in the first iteration. However, since a sine wave is symmetric from quadrant to quadrant, every sine value from 0 to 2π can be represented by reflecting and/or inverting the first quadrant appropriately.

III. TAYLOR SERIES APPROXIMATION

One way to find the sine, cosine of a given angle is to take a certain amount of terms (the more terms we take, the more accurate the approximation) from the Taylor series[3]. A Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point. If the function *f(x)* possesses continuous derivatives of orders 0, 1, 2,….*(n* + 1*)* in a closed interval *I* = [*a*,*b*], then for any *c* and *x* in *I* , it can be approximated by using a finite number of terms of its Taylor series.

$$f(x) \approx f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \frac{f'''(c)}{3!}(x-c)^3 + \cdots \quad (6)$$

$$f(x) \approx \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!}(x-c)^k$$

Here, rather than using =, we have written ≈ to indicate that we are not allowed to assume that *f(x) equals* the series on the right. Series (6) is called the **"** *Taylor series of f at the point c.***"**

In the special case *c* = 0, series (6) is also called a *Maclaurin series***.**

$$f(x) \approx f(0) + f'(0)\cdot x + \frac{f''(0)}{2!}\cdot x^2 + \frac{f'''(0)}{3!}\cdot x^3 + \cdots \quad (7)$$

$$f(x) \approx \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}\cdot x^k$$

It is interesting to note that, provided we add infinitely many terms, we have not just an approximation, but an equality for most functions *f*(*x*). This means that we can make our approximation as good as we like by adding enough terms. In practical computations with Taylor series, it is usually necessary to *truncate* the series because it is not possible to carry out an infinite number of additions. A series is said to be *truncated* if we ignore all terms after a certain point.

Taylor series expanded about c = 0 for sine and cosine function are the following

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!}\cdot x^{2k+1} \qquad (|x| < \infty) \quad (8)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!}\cdot x^{2k} \qquad (|x| < \infty) \quad (9)$$

IV. IMPLEMENTATION AND RESULTS

In this paper, we have presented the implementation of CORDIC and Taylor algorithm in a ARM Cortex MO, PSoC4, CY8C4245AXI-483 architecture (Fig. 2). In this article are used 32 iterations to calculate the sine and cosine values in the fixed-point version with CORDIC algorithm. Using 32 iterations, values have a high precision and the increasing step is 1°. The processing of sine and cosine with a 9 terms Taylor algorithm is in the 32 bit single precision floating point version.

By verification of accuracy, was concluded that there is a accuracy equivalent in the calculated sine and cosine values with the Cordic and Taylor algorithms.

Table 1 shows the results of the implementation of the CORDIC and Taylor algorithm on the device.

Table 1

Methode	Memory used		Time execution
	Flash	SRAM	
CORDIC	24.0%	58.9%	0.023656
Taylor	77.3%	57.5%	0.056059

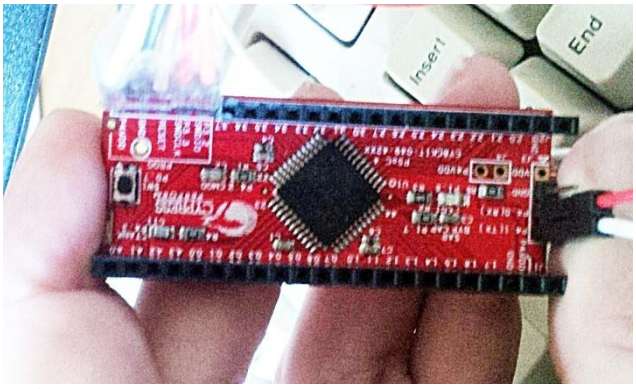

Fig 2. ARM Cortex MO, PSoC4, CY8C4245AXI-483 processor.

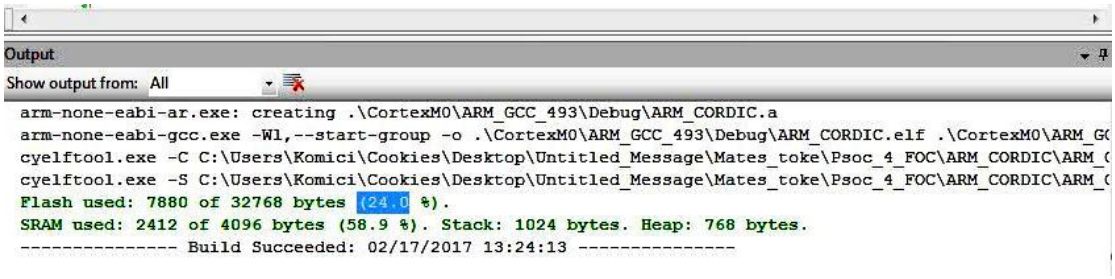

Fig 3. Analysis of memory used by CORDIC algorithm.

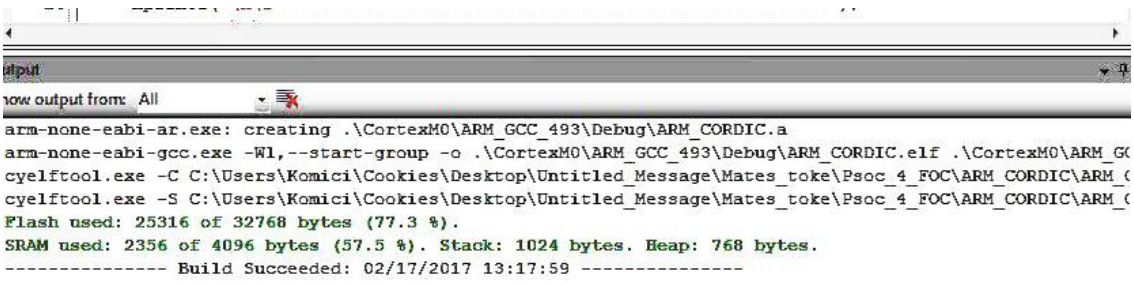

Fig 4. Analysis of memory used by Taylor algorithm

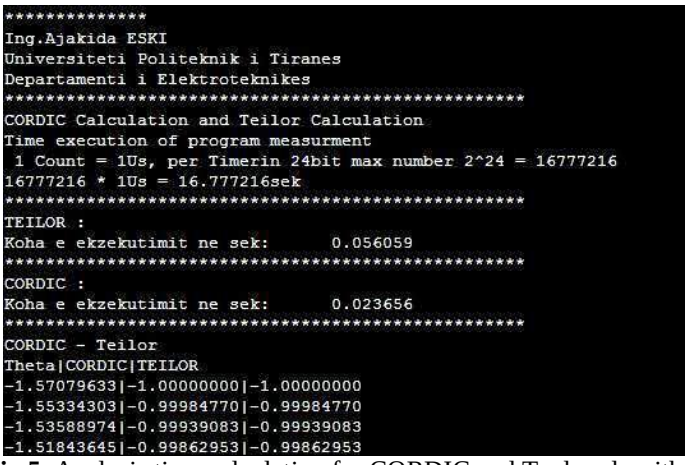

Fig 5. Analysis time calculation for CORDIC and Taylor algorithm

V. CONCLUSION

Using a method for rapid evaluation of sine and cosines, become more and more serious when performing real-time digital signal processing. A detailed comparison between the CORDIC and Taylor algorithm for the calculation of the sine and cosine functions have been evaluated. Through Table 1 we prove that, the CORDIC algorithm can calculate 2.3 times faster and occupies 47.1% less memory than Taylor algorithm, although unlike it calculates the sine and cosine values at the same time. This means a smaller and cheaper design, a smaller power consumption. It is clear that a architecture of 32 bits CORDIC algorithm can successfully be applicable to the real time operation.

REFERENCES

[1]. E. Volder, The CORDIC Trigonometric Computing Technique, *IRE Trans. Electronic Computers*,vol. EC-8, no. 3, 1959, 330–334

[2]. Bimal Gisuthan, Thambipillai Srikanthan, Pipeline flat CORDIC based trigonometric function generators, *Microelectronic Journal*, *33*, 2002, 77-89.

[3]. Ward Cheney, David Kincaid, Review of Taylor series, *Numerical Mathematics and computing*, (Thomson Higher Education, 2008) 20-27.

[4]. Javier Valls, Martin Kuhlmann, Keshab K.Parhi, Evaluation of CORDIC Algorithm for FPGA Design, *Journa of VLSI signal Processing*, 32, 2002, 207-202