FPGA real-time calculator to determine the position of an emitter

M.Tamura＊, T. Aoyama
The Faculty of Engineering, Miyazaki University, Japan
Gakuen Kibanadai-Nishi 1-1, Miyazaki 889-2192, Japan
(Fax: +81-0985-58-7411; *E-mail: tgb321u@student.miyazaki-u.ac.jp)

Abstract: To detect motions of bodies, we have discussed them with two viewpoints; one is a detection algorithm, and another is the hardware implementation. The former is to find small terms expansions for sine/cosine functions. We researched Maclaurin and optimum expansions, and moreover to reduce hardware amounts, revised the expansions. The expansions don't include divide calculations, and the error is within 0.01%. As for the former problem, there is another approach also; that is the cordic method. The method is based on the rotation of a vector on the complex plain. It is simple iterations and don't require large logic. We examined the precision and convergence of the method on C-simulations, and implemented on HDL. The later problem is to make FPGA within small gates. We considered approaches to eliminate a divider and to reduce the bit number of arithmetic. We researched Newton-Raphson's method to get reciprocal numbers. The higher-order expression shows rapid convergence and doesn't be affected by the initial guess. It is an excellent algorithm. Using them, we wish to design a detector, and are developing it on a FPGA.

Keywords: FPGA, HDL, cordic, divider, reciprocal numbers, sine-function, cosine-function, optimum expansion

1. INTRODUCTION

It is meaningful that determines equation of motions on real time, and predicts the locations of objects in future. Where, we must determine the locations currently; it is a necessary condition. We believe that the limitation of software means would become conspicuous. We pay attention to Hardware Descriptor Language (HDL) and Field Programmable Gate Array [1] (FPGA). They are techniques that make layout of field-programmable LSI. Using them, we can plan and design small scale computing resources. However, they must be designed under some restrictions; i.e., the number of bits, elimination of complex functions, annihilation of division, and so on. Our objective is to avoid these restrictions by developing new techniques and to equip them on FPGA.

LSI implementations of complex functions are interesting research, which causes deep understanding of algorithms. In this paper, we try to develop one-chip calculator to determine the position of an emission source. It is development of means to find status in a space; and it is also a step for non-destruction examinations.

We defined the chip's specifications as followings.
(1) There is a space, which includes an object. The object emits radiation randomly. Increasing the distance from a source, the radiation intensity decreases toward zero non-linearly.
(2) Two detectors are set to detect the radiation. The detector's locations are fixed beforehand.
(3) On such a system, we wish to determine current positions of a moving object, and want to make the size of a calculation-unit minimize.
(4) We wish to shorten the developing terms and costs. Thus, we adopted a FPGA of 20[KG].

2. DETERMINATIONS OF LOCATIONS

Some determination methods for current locations of moving objects are known. A general procedure is based on the triangulate; that is, we define a triangle (figure 1).

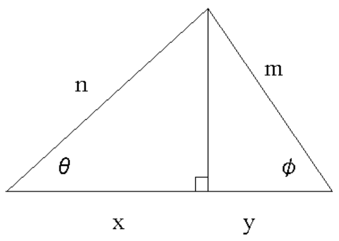

Fig 1. Geometry of an object and two detectors.

The problem is defined as "evaluate unknown variables x, y, m, and under the following conditions."

$L=x+y,$ (1)
$x=n*\cos(\theta),$ (2)
$y=m*\cos(\phi),$ (3)
$n*\sin(\theta)=m*\sin(\phi).$ (4)

Now, we show a scheme to evaluate the variables.

From eqs. (1-3), we get,

$n*\cos(\theta)+m*\cos(\phi)=L.$ (5)

So, we obtain "n" as,

$n=\{L-m*\cos(\phi)\}/\cos(\theta).$ (6)

Then, we get,

$m=\{\sin(\theta)/\sin(\phi)\}\{L-m*\cos(\phi)\}/\cos(\theta),$ (7)

$m[1+\{\sin(\theta)/\sin(\phi)\}\{\cos(\phi)/\cos(\theta)\}]$
$=\{\sin(\theta)/\sin(\phi)\}\{L/\cos(\theta)\}.$ (8)

Therefore,

$m=\{\sin(\theta)/\sin(\phi)\}\{L/\cos(\theta)\}/[1+\{\sin(\theta)/\sin(\phi)\}\{\cos(\phi)/\cos(\theta)\}]$
$=\{L/\cos(\theta)\}/[\{\sin(\theta)/\sin(\phi)\}^{-1}+\{\cos(\phi)/\cos(\theta)\}].$ (9)

Considering the right side of eq. (9), the major term is $1/\cos(\theta)$. The term controls a distance between an object and two detectors. If the object is separate from the detectors, the argument is $\theta \rightarrow \pi/2$. On the limit, high precision calculation of $1/\cos(\theta)$ is required. For the calculation, we adopt,

$1/\cos(\theta)=1/\sin(\pi/2-\theta).$ (10)

We rewrite $\pi/2-\theta=z$, and,

$1/\sin(z),\ z\rightarrow 0.$ (11)

The term can be evaluated by Maclaurin expansion. Thus, for infinitesimal small z, we get,

$\sin(z)=z-z^3/6+z^5/120.$ (12)

$1/\sin(z)=1/(z-z^3/6+z^5/120)=(1/z)+R.$ (13)

The eq. (13) is so accurate; in an interval, $0<z<\pi/64$, we get,

$1/\sin(z)=(1/z)+z/6.$ (14)

The precision of eq. (14) is O(-7). The eq. (14) must be implemented by HDL and FPGA; it is not so easy problem. The term 1/z cannot be expanded by polynomials but be expressed by iterations. The expression will be discussed in section 4.

The term, $\cos(\phi)/\cos(\theta)$, can be rewrite by relation, $\phi=\theta+\varepsilon<\pi/2$; therefore, the term becomes 1/1 on the limit. Thus, there is no problem for the precision.

The term, $1/\{\sin(\theta)/\sin(\phi)\}$, is;

$1/\{\sin(\theta)/\sin(\phi)\}=\sin(\phi)/\sin(\theta)=\sin(\theta+\varepsilon)/\sin(\theta)$
$=\{\sin(\theta)\cos(\varepsilon)+\cos(\theta)\sin(\varepsilon)\}/\sin(\theta)$
$=\cos(\varepsilon)+\sin(\varepsilon)/\tan(\theta)\sim\cos(\varepsilon)\sim 1.$ (15)

Therefore, there is no problem. So, we get,

$m=\{L/\cos(\theta)\}/[\{\sin(\theta)/\sin(\phi)\}^{-1}$
$+\{\cos(\phi)/\cos(\theta)\}]\rightarrow L/\{2\cos(\theta)\},$ (16)

and we get for variable "n",

$n=[L-\{\sin(\theta)/\sin(\phi)\}\{L/\cos(\theta)\}/[1+\{\sin(\theta)/\sin(\phi)\}\{\cos(\phi)/\cos(\theta)\}]*\cos(\phi)]/\cos(\theta).$ (17)

Here, we check an influence of small bit number.

In case of $\theta=\phi$, eq. (16) is,

$m=(1/2)\cos(\theta).$ (18)

We calculated eq. (18) in 10-bits numerical calculations, and got figure 2.

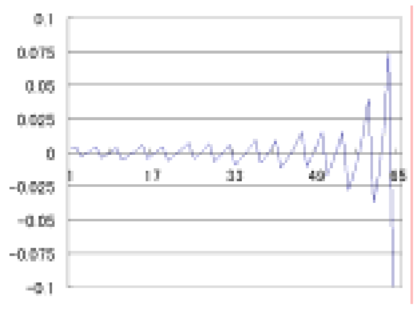

Fig 2. Error curve in case of 10-bits calculations for eq.(18).

The point "1" on horizontal axis is 87.19 [deg], and point "65" is 89.96 [deg]. The notch on horizontal axis is 0.04327[deg]. In the region $\theta<89.44$ [deg], the error of distances between object and detector is lower than unit length L=1.

On $\theta=89.44$[deg], the distance is ~51; therefore, the calculation error is lower than (+/-)0.02, until 50 times far of unit length. The results are got as ability of the algorithm. If we used 10-bits calculations, the resolution is $1/2^{10}=0.00098\sim O(-3)$. The discrete error is 20 times smaller than that of the algorithm error.

3. FPGA EXPRESSIONS OF TRIANGULAR FUNCTION

When triangular functions are implemented on FPGA, various algorithms are known [1]. We researched them on viewpoints, hardware amounts, precision, annihilation of divide, and so on.

3.1 MaClaurin expansions

Maclaurin expansions for triangular functions are,

$\sin(x)=x-x^3/6+...,$ (19)

$\cos(x)=1-x^2/2+x^4/24-....$ (20)

Here, we calculated them in interval $0<x<\pi/2$.

In eqs.(19, 20), there is no divide calculation. Constants, 1/6 and 1/24, are executed as multiply of finite decimals. If we use 32-64 bits floating-point numbers, eqs. (19, 20) are practical in interval $0<x<\pi/4$.

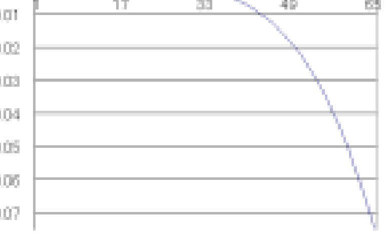

Fig. 3 Precision of $\sin(x)=x-x^3/6$.

The point "1" on horizontal axis is 0 [deg], and point

"65" is 90 [deg]. The notch on horizontal axis is 1.406 [deg]. In region $\pi/4<x$, the expression has not practical accuracy.

3.2 Optimized expansions

Optimized expansions [2,3] replaced by power series of 2.

Optimized expansions are excellent algorithm; however, they require floating-point numbers more than 32-bits. They may be developed for integrated functions in mathematical library. The hardware is larger amount. We wish to reduce them without decline of the precision. We tried some algorithms and examined the characters under the following conditions.

(1) Replace coefficients to power of 2.

(2) The highest power of x is set to be 3.

(3) Revise the precision of 2-terms Maclaurin expansion, $\sin(x)=x-x^3/6$.

(4) Expand effective region to be over $\pi/4$.

Finally, we get an expression; that is,

$\sin(x)=x-x^2/32-x^3/8$. (21)

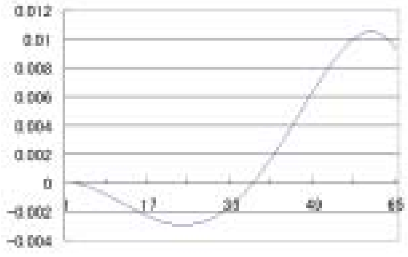

Fig. 4 Precision of $\sin(x)=x-x^2/32-x^3/8$.

The point "1" on horizontal axis is 0 [deg], and point "65" is 90 [deg]. The notch on horizontal axis is 1.406 [deg].

In whole region, calculation error is almost lower than 1%. The precision is 7 times high compared with original Maclaurin expansion (figure 3), and moreover the coefficients are calculated by a shifter. We also found an expression, $\sin(x)=(1-1/512)x-x^2/32-x^3/8$.

This expansion gives error under 0.8% in whole region; however, this gives low precision nearby x=0. Therefore, we didn't use it.

3.3 Cosine expoansions

Mathematically, cosine function is same characters for sine functions. However, on the optimized expansions, the both functions are not equal. We tried the precision of cosine function expansions. That is, we selecte an simple expression, $\cos(x)=1-x^2/2$, and calculate it. The results are shown in figure 5.

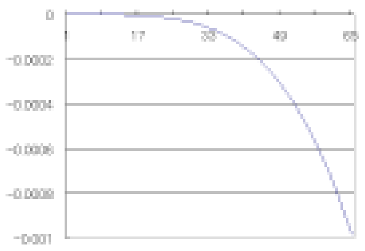

Fig. 5 Precision of $\cos(x)=1-x^2/2$.

The point "1" on horizontal axis is 0 [deg], and point "65" is 90 [deg]. The notch on horizontal axis is 1.406 [deg].

In whole region, calculation error is lower than 0.1%. The expression is very simple one; however, the precision is 10 times high compared with sine optimized expansion (figure 4). The expansion has high precision extremely, error=0.02%, in interval $x<\pi/4$.

We got more accurate expression, $\cos(x)=1-x^2/2+x^3/64$; the precision is following.

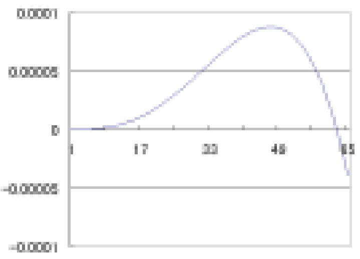

Fig. 6 Precision of $\cos(x)=1-x^2/2+x^3/64$.

The point "1" on horizontal axis is 0 [deg], and point "65" is 90 [deg]. The notch on horizontal axis is 1.406 [deg].

In whole region, calculation error is lower than 0.01%. The precision is 100 times high compared with sine optimized expansion (figure 4). Moreover, the error near $x=\pi/2$ is low; this is an useful character.

3.4 Cordic algorithm

The cordic algorithm [4] emulates the function value as a vector on the complex plane. It rotates the vector by operating a complex number, and after the iterative rotations, it gives the function value as the real-part of the vector. The values of sine, cosine, co-tangent, hyperbolic-sine, hyperbolic-cosine, exponential, and logarithm functions can be obtained by the algorithm by changing of an initial value and the convergence conditions. The algorithm can be implemented in a small-scale circuit comparatively. So, scientific-calculators often build in it. It is a notified algorithm; so, we researched the realization.

To simplify the discussions, we limit to sine and cosine functions. The outline of the algorithm is followings.

It thinks about a unit vector ***f*** with the argument θ on a starting point. We write the real and imaginary parts as x and y.

$$\begin{cases} x = \cos\theta \\ y = \sin\theta \end{cases} \qquad \begin{cases} x^2 + y^2 = 1 \\ \theta = \tan^{-1}\dfrac{y}{x} \end{cases}$$

Here, an initial guess is θ=0, x=1, and y=0. It is assumed that target phase angle t is given as an argument of the requested function. It thinks making it to $\theta = 0 \to t$ by rotating vector ***f***. Here, the operator where the angle α is rotated to the vector on the complex plane will be defined, and it be called correction vector ***g***. This conversion can be described as follows by the procession form.

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & p\delta \\ -p\delta & 1 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \boldsymbol{g} \cdot \boldsymbol{f}$$

Parameter p decides the direction of conversion (direction of the rotation). And it is 1or-1. The absolute value of vector f is done by this conversion in the $\sqrt{1-\delta^2}$ time.

Here, the parameter δ is provided as follows.

$\delta = 2 - (n-1) = 1, 1/2, 1/4, 1/8 \cdots$

We show the expression of the change in the phase angle as follows.

$$\theta_{n+1} = \theta_n + \alpha \quad (\alpha = \tan^{-1}(\delta))$$

After all, this conversion and multipling vector g that length is $\sqrt{1-\delta^2}$, and the phase angle is α is same. Sign p in the direction of the phase angle degree correction is made a sign opposite to $\theta - t$. This vector rotation is repeated until error margin $(\theta - t)$ becomes small enough. The absolute value of function vector f leaves one, and is finally as follows.

$$|\boldsymbol{f}| = \sqrt{1+\left(\frac{1}{1}\right)^2} \times \sqrt{1+\left(\frac{1}{2}\right)^2} \times \sqrt{1+\left(\frac{1}{4}\right)^2} \times \cdots \times \sqrt{1+\left(\frac{1}{2^{n-1}}\right)^2}$$

$$\approx 1.646760258\cdots$$

This value steadies without any relation to t.

$$|\boldsymbol{f}| = \frac{1}{1.646760258\cdots} = 0.60725\cdots \qquad \boldsymbol{f} = 0.60725\cdots + j0$$

Then, the absolute value of an initial value of f is assumed to be this reciprocal, and it is assumed initial phase $\theta = 0$. We get the last settling result.

A trigonometric value was obtained by becoming real part is cos t and imaginary part is sin t of f.

3.4.1 Verification of accuracy of cordic algorithm

We verified accuracy of the cordic algorithm by using C program. The objects are;

(1) dependency of iterations for accuracy,

(2) dependency of significant digits,

(3) number of iterations to get precision of 3-digits.

Concerning to (1), we simulated the codic algorithm for sine-function. Thus, we calculated sine-function values from 0 to π/2, on 10-25 iterations. We examined the limit of error for average and maximum ones.

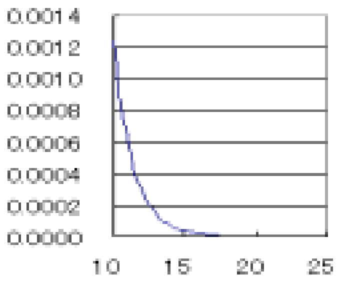

Fig. 7 Limit of average error.
Horizontal axis is the number of iterations. The numbers on vertical axis show limit of average error for various arguments.

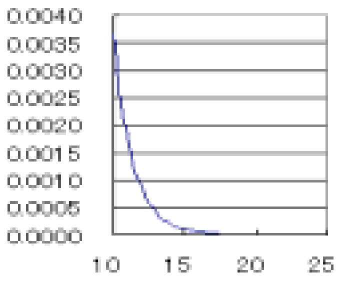

Fig. 8 Limit of maximum error.
Horizontal axis is the number of iterations. The numbers on vertical axis show limit of maximum error for various arguments.

Concerning to (2), we simulated sine-functions for some digits, that is 1 to 5 significant digits. The arguments of the sine-function are 0 to π/2. The number of iterations is 25. We examined the limit of error for average and maximum. We adopted the following approximation for the test. The approximation C-language-cords are following;

```
a = x * F + 0.5;
x = a;
x = x / F;
```

Where "a" is an integer variable of 4B, and "F" is a IEEE-real variable of 8B. We use Fs in interval [1,10000], by decreasing of calculation-precision, simulate the movements.

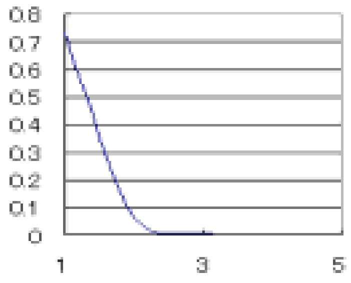

Fig. 9 Limit of average error.
Horizontal axis is plotted by significant digits in the decimal. The numbers on vertical axis show limit of average error for various arguments of sine-function.

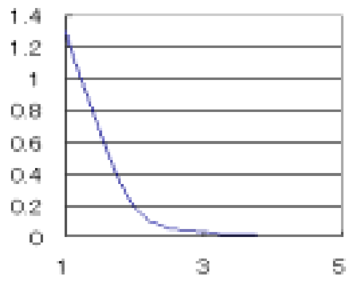

Fig. 10 Limit of maximum error.
Horizontal axis is plotted by significant digits in decimal. The numbers on vertical axis show limit of maximum error for various arguments of sine-function.

Concerning to (3), we simulated the cordic algorithm for sine-function. The simulations were executed from arguments 0 to $\pi/2$, and on 7-14 iterations. Whole calculations were done in 3 decimal digits. We examined the average and maximum errors.

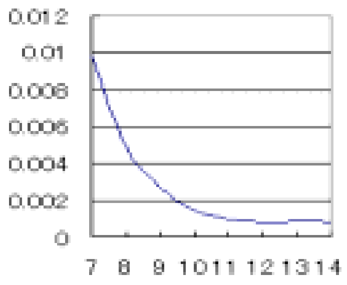

Fig. 11 The average error-limits
The scales on horizontal axis are the iteration number. Vertical axis shows average error.

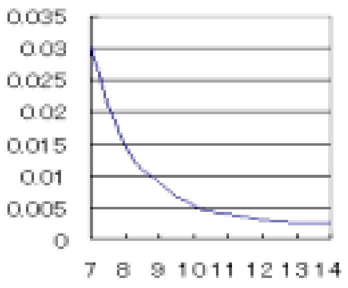

Fig. 12 Limits of maximum error.
Horizontal axis is the iteration number. Vertical axis indicates maximum error. On 12 iterations, the average error is low the resolution of decimal 3 digits.

Until now, we considered the processing in the decimal 3 digits. To optimize it, hereafter we treat all in binary. So, we can use multiply and right-shift operations instead of multiply and divide ones. In FPGAs, elimination of the division reduces hardware amounts. The precision of decimal 3-digits is equivalent to binary 10-bits. To test the binary case, we replaced “F=1000” to “F=1024” [see section 3.4.1 concerning (2)]. The results are listed in figure 13 and 14.

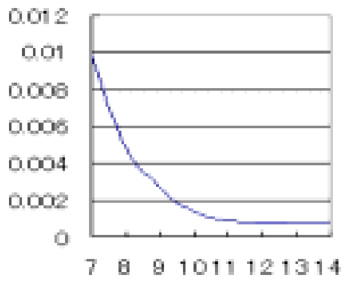

Fig. 13 Limits of average error.
Horizontal axis is the iteration number. Vertical axis indicates average error.

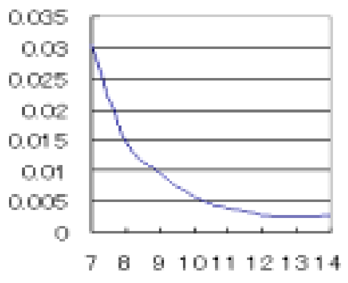

Fig. 14 Limits of maximum error.
Horizontal axis is the iteration number. Vertical axis indicates maximum error.

4. ELIMINATION OF DIVIDER: RECIPROCAL ITERATIONS

We discuss a calculation method of 1/sin(z), z~0. It cannot be executed by expansions. Usually, Newton-Raphson method is used. It is iterations and requires an initial guess. It is difficult to find appropriate guesses. However, the function, 1/z, is not so irregular one; therefore, we expect rapid convergence of Newton-Raphson, and try some higher-order expressions. They are followings.

2nd. order: $Xn+1 = 2Xn - z * Xn^2$, (22)

3rd. order:

$h=1 - z * Xn$, $Xn+1 = Xn * (1 + h + h^2)$, (23)

4th. order:

$h= 1 - z * Xn$, $Xn+1 = Xn * (1 + h)(1 + h^2)$, (24)

Where, we set initial guess X0=1 for all z-values.
When z=0.01, the error covergences of eqs. (22-24) are plotted in figure 15.

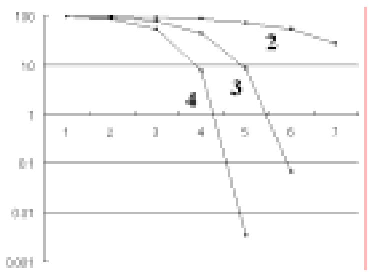

Fig. 15 Error convergence of Newton-Raphson method. Vertical axis is logarithm scale [100-0.001]. Horizontal axis is the number of iterations. Digits "2-4" are second, third, and fourth order's expressions, respectively. Whole initial guess is 1.

The guess is not appropriate value; therefore, rapid convergence character of higher-order is required. We tried the fourth order expression for many values of z. As the results, to get error of O(-4), the maximum number of iterations was 9. Thus, if we used the expression, we might omit dividers in FPGA.

CONCLUSIONS

Our objective was to design a FPGA that detected motions of bodies in real-time processing. Now, we are at the stage that simulations of triangular functions are completed. We examined some optimal expansions and characters of the cordic algorithm.

The new expansions include polynomials until 3rd order and have coefficients of power of 2. The evaluations don't require divisions, and the approximation level is 7 times higher than the same order of Maclaurin expansions.

For the cordic algorithm, we researched numerical-representation bits and iteration-number to realize resolution of O(-3) for function evaluations. Those were 9-10 bits in binary, and 12 iterations. We are sure that the cordic algorithm is useful. We are now constructing of it on a FPGA.

Then, we can calculate the locations of moving bodies within O(-3). When they are very far from the detectors, the high precision evaluation of "sin(z), z~0" is required. The evaluation is equivalent to high precision calculation of the reciprocal numbers. It cannot be processed by expansions or the cordic algorithm; so, we examined Newton-Raphson's method. The higher order expressions have less dependency for initial guesses, and moreover, it shows rapid convergence. On 5-9 iterations, O(-7) approximation is got. Thus, we can predict the locations in case of very far area. Since the expressions are very useful, we have no longer equipped dividers on FPGA, as well as "3D-Now".

REFERENCES

[1] Xilinx Inc. Home page: "Programmable Logic Devices, FPGA & CPLD",
URL: http://www.xilinx.com/

[2] J.Yamauchi, T.Uno, and S.Hitotsumatsu, "Numerical Calculations for computers (in Japanese)", Vol.3,
Baifu-kan Pub. Co. Ltd.(1972,Tokyo).

[3] M.Mori, "FORTRAN 77 Programming for Numerical Calculations (in Japanese)",
Iwanami Pub. Co. Ltd.(1986,Tokyo),
ISBN4-00-007684-1.

[4] T. Srikanthan and B. Gisuthan
"A novel technique for eliminating iterative based computation of polarity of micro-rotations in CORDIC based sine-cosine generators"
Volume 26, Issue 5 , 10 June 2002