Hardware of the Elementary Function by CORDIC

Naohiro Iwanaga*, Hiroyuki Tanabe, Akira Yamawaki

Kyushu Institute of Technology, Fukuoka, Japan(804-8550)

*Corresponding Author. serikawa@elcs.kyutech.ac.jp

Abstract

It sets to image processing, Usually, Elementary-function operations, such as trigonometric functions and logarithm, are performed. In hardware of image processing, I think that it is useful with the point that hardware the elementary function of these can reduce design loads called shortening of development time. Then, an algorithm called CORDIC was used and processing of sin and cos which are trigonometric functions was fixed-point. A result which can say that the arithmetic precision is equivalent to math.h which is a standard header file of the C language was brought.

Keywords: CORDIC, FPGA, VHDL

1. Introduction

With advanced features of inclusion apparatus, hardware of image processing is becoming indispensable. And elementary-function operations, such as trigonometric functions, an exponential function, a logarithmic function, are carried out to image processing in many cases. When performing image processing at high speed, improvement in the speed of elementary-function operation becomes indispensable. Then, although processing can be accelerated by using an algorithm called CORDIC as the example and such a hardware library exists, neither performance nor a circuit scale is clarified, and it is difficult for other developers using it. In this research, in various elementary functions, it is flexible, and hardware processing of sin and cos using an algorithm called CORDIC which is simple processing using a table. In this paper, processing of sin and cos is fixed-point, and it is shown that the arithmetic precision is equivalent to the single precision floating point version. Furthermore, processing time, a circuit scale, and

clock frequency are clarified through the mounting experiment to FPGA.

2. The hardware design of trigonometric functions

2.1 CORDIC

First, an algorithm called CORDIC is an algorithm which performs elementary-function operation. It is the repetitive solving method for the ability to obtain the solution convergent by considering a function to be a vector on a two-dimensional plane, and carrying out repetitive calculation of the rotation of a vector. Furthermore, as the preceding clause showed, in various elementary functions, it is flexible. The definitional equation of CORDIC is shown in (1) of the following, (2), and (3).

$$x[j+1] = x[j] - m\sigma_j 2^{-j}y[j] \quad (1)$$

$$y[j+1] = y[j] + \sigma_j 2^{-j}x[j] \quad (2)$$

$$z[j+1] = \begin{cases} z[j] - \sigma_j \tan^{-1}(2^{-j}) & \text{if } m = 1 \\ z[j] - \sigma_j \tanh^{-1}(2^{-j}) & \text{if } m = -1 \\ z[j] - \sigma_j (2^{-j}) & \text{if } m = 0 \end{cases} \quad (3)$$

x and y express an x-coordinate and y coordinates, respectively, and z is an angle when it indicates by polar coordinates. It changes by whether a vector rotates σ_j to positive direction, or it rotates in the negative direction, and when rotating to positive direction and rotating in 1 and the negative direction, it becomes a value of -1. m differs in a value according to the kind of function. When m is 1, coordinates like sin or cos are a function which changes in the shape of a circle, and when m is -1 and the function and m from which coordinates like sinh or cosh change in the shape of a hyperbola are 0, coordinates are a function which carries out alignment change. Concretely, the example of a function realizable with these values is shown in the

following Table 1. Although Mode of Table 1 has two, rotation and vectoring, this is the difference in the method of convergence. In Rotation, the angle Z is completed as the input $Z_{in}(\theta)$ by rotation of a vector, and, in vectoring, y coordinates are completed as 0 by rotation of a vector.

Table1. The example of a function by CORDIC

M	Mode	Initial values			Functions	
		X_{in}	Y_{in}	Z_{in}	X_R	Y_R or Z_R
1	rot	1	0	θ	$\cos \theta$	$\sin \theta$
-1	rot	1	0	θ	$\cosh \theta$	$\sinh \theta$
-1	rot	a	A	θ	ae^{θ}	ae^{θ}
1	vec	1	A	$\frac{\pi}{2}$	$\sqrt{a^2 + 1}$	$\cot^{-1}(a)$
-1	vec	a	1	0	$\sqrt{a^2 - 1}$	$\coth^{-1}(a)$
-1	vec	$a + 1$	$a - 1$	0	$2\sqrt{a}$	$0.5 \ln(a)$
-1	vec	$a + \frac{1}{4}$	$a - \frac{1}{4}$	0	$\sqrt{a}$	$\ln(\frac{1}{4}a)$
-1	vec	$a + b$	$a - b$	0	$2\sqrt{ab}$	$0.5 \ln(\frac{a}{b})$

Thus, CORDIC is excellent in the point that various elementary functions are realizable, by changing m and Mode. Next, the formula which asks for the angle θ_j required in order to rotate a vector with this algorithm is shown in following (4).

$$\theta_j = \tan^{-1}(2^{-j}) \quad (4)$$

This angle θ_j is repeated and it asks according to the number of times. Since a solution can be converged by repeating at least 17 times, from θ_0 to θ_{17} is used in fact. θ_j to be used is shown in the following Table 2.

Table2. The angle for θ_j

j	$\theta_j = \tan^{-1}(2^{-j})$
0	45
1	26.56505118
2	14.03624347
3	7.125016349
4	3.576334375
5	1.789910608
6	0.89517371
7	0.447614171
8	0.2238105
9	0.111905677
10	0.055952892

11	0.027976453
12	0.013988227
13	0.006994114
14	0.003497057
15	0.001748528
16	0.000874264
17	0.000437132

About the accuracy of convergence, the arbitrary angles from 0 [deg] to 90 [deg] can be denoted by sufficient accuracy with the combination of addition and subtraction from θ_0 to θ_{17} . Moreover, after converging a vector using Table 2, a formula (1), (2), and (3) about X_R and Y_R of Table 1, division is done by R of the absolute value of the vector. As an example, the right triangle in $j=0$ of Table 2, 1, and 2 was shown in (a) of Fig. 1, and the absolute value of the vector changed into (b) of Fig. 1 was shown.

(a) The right triangle used as θ_j

(b) The absolute value of a vector

Fig 1. Right triangle used by CORDIC

Only the case of $j = 0, 1$, and 2 is illustrated in Fig. 1. However, when calculating X_R and Y_R of Table 1 with the elementary function derived this time, the absolute value of the conversion vector in $j = 17$ is required. In practice, X_R and Y_R can be calculated by doing division of the X coordinate and y coordinate for which it asked by the formula (1) and (2) by R_{17} .

2.2 Design of Hardware

The data path of hardware is shown in Fig. 2 in the hardware design of sin and cos using a CORDIC algorithm. Although the algorithm of the preceding clause was hardware obediently fundamentally, some initial values, number of times of a repetition, etc. were changed.

Fig2. The data path of hardware

By this hardware, in the state S1, 16 bit shifts of the input angle deg are carried out to the left, and it is set to Z1.

In the state S2, count +1 bit shift of $X0$ and $Y0$ is carried out to the right, and it is set to XX and YY. Moreover, only 1 will be added if count will be in the state S4.

In the state S3, $X0$ and YY, $Y0$ and XX, $Z0$, and arctan are subtracted and added, and each calculation result is set to $X1$, $Y1$, and ZZ by performing size comparison of Z1 and $Z0$. In addition, arctan changes with count and a value is the same value as the table for convergence of Table 2.

In the state S4, $X1$, $Y1$, and ZZ which were calculated by S3 are set to $X0$, $Y0$, and $Z0$, respectively, and one is further added to count.

If count is set to 16, it will change in the state S5, and except it, it returns to S2 and calculates to S4 again.

In the state S5, 32768 is added to $X0$ and $Y0$,

respectively for rounding off, and it is set to $X1$ and $Y1$

In the state S6, $X1$ and $Y1$ are multiplied by r , and calculation of cos and sin is completed.

3. Experiment/Results

It verified by measuring the accuracy of the developed fixed-point version with the processing result of sin and cos of math.h which is a standard header file of the C language. However, since the decimal fraction of the output of the developed program was calculated as 24 bits, it compared with the processing result of math.h by multiplying by 2^{24} . The error by an input angle is shown in the following Table 4.

Table4. The error by an input angle

deg	sin	cos
10	0.39[%]	0.17[%]
20	0.05[%]	0.17[%]
30	0.06[%]	0.16[%]
40	0.11[%]	0.14[%]

It can be said that it is equivalent to the accuracy of the fixed-point version developed from this result, and math.h.

Next, designed hardware was mounted in SPARTAN-6 of Xilinx. The result, It turned out that the clock period can operate by 6.074 [ns]. A circuit scale is shown in Table 5.

Table5. The circuit scale at the time of mounting

	Used	Available	Utilization
Registers	242	18224	1%
LUTs	510	9112	5%

By this hardware, since it processed using the multiplier without using the division machine, the number of a Registers and LUTs were able to be lessened.

4. Conclusions

The method of fixed-point-izing at the time of making a CORDIC algorithm into hardware and the details of hardware were shown. By verification of accuracy, it checked that there was accuracy equivalent to the value of

math.h, and the details of the circuit scale were shown. From now on, performance gain by pipeline processing is performed and comparison of a circuit scale with model processing or a clock period is performed one by one. Moreover, Hardware of elementary functions other than trigonometric functions is verified. Furthermore, the library of the elementary function by CORDIC is created. I think that what is necessary is just to be able to provide for a researcher.

Acknowledgment

In advancing this research, polite and eager guidance was received from Assistant Professor Akira Yamawaki. Thank you very much.

References

- (1) Miloš D. Ercegovic and Tomáš Lang : “DIGITAL ARITHMETIC”, Elsevier Science, Volume 11, 2003
- (2) B.Lakshmi and A.S.Dhar : “CORDIC Architectures: A survey”, Hindawi Publishing Corporation, Volume 2010, ArticleID 794891, 19pages
- (3) J. S. Walther , “A unified algorithm for elementary functions”, Hewlett-Packard Company, Palo Alto, California
- (4) Jack E. Volder : “The CORDIC Trigonometric Computing Technique”, IRE Trans Electron Comput, 1959