Research Journal of Applied Sciences, Engineering and Technology 6(1): 119-122, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: October 30, 2012 Accepted: December 21, 2012 Published: June 05, 2013

An FPGA Implementation of the Natural Logarithm Based on CORDIC Algorithm

Shaowei Wang, Yuanyuan Shang, Hui Ding, Chen Wang and Junming Hu
College of Information Engineering, Capital Normal University, Beijing 100048, China

Abstract: In digital signal and image processing, it's very common to calculate the value of certain transcendental functions, such as natural logarithmic function. This study introduces the basic principles of the mode of calculation of the hyperbolic systems by using the CORDIC algorithm, then analyses the Field-Programmable Gate Array (FPGA) CORDIC core processing unit in detail. The biggest advantage of the CORDIC algorithm is that its circuit structure is very simple, using only adder and shifter. It is very suitable for FPGA implementation. Based on the iterative algorithm, a FPGA implementation of the natural logarithmic function has been designed. The pipelined-FPGA architecture can achieve a high computational speed, for completing a computation only requires one clock cycle. The relative error values are below 10^{-4}, which can satisfy the accuracy requirements.

Keywords: CORDIC, field-programmable gate array, natural logarithmic, pipelined

INTRODUCTION

The natural logarithmic function is widely used in image processing and digital signal processing. For example, to solve some questions in the physical setting of logarithmic imaging processes, Michel and Jean-Charles (1995) used the Logarithmic Image Processing (LIP) model, known to be a compatible mathematical framework. After recalls on the model, that study presents two images transforms: one performs an optimal enhancement and stabilization of the overall dynamic range and the other does of the mean dynamic range. What's more, a new method based on gray-natural logarithm ratio bilateral filtering is presented for image smoothing in Guannan Chen's work (Guannan *et al.*, 2008). Therefore, the study of its hardware implementation is necessary. The logarithm is widely implemented with: Table lookup method, Taylor series expansion method and linear approximation method. With the improvement of the accuracy requirements, the table lookup method will consume a lot of resources. Taylor series expansion method will use the multiplier. As precision requirements increase, the demand of the multiplier will become larger. However, the number of multipliers in the FPGA is relatively limited. Linear approximation method can't meet the high precision requirements.

The biggest advantage of the CORDIC algorithm is its circuit structure is very simple, using only adder and shifter. As a hardware efficient algorithm, it is particularly suitable for FPGA implementation. In order to avoid the complexity of the circuit, the idea of the CORDIC algorithm is that decompose the rotation operation into successive basic rotations and every rotation only be realized with add-and-shift operations.

CORDIC ALGORITHM PRINCIPLE

The CORDIC (Coordinate Rotation Digital Computer) algorithm was first introduced by Volder (1959). It is efficient to compute the values of trigonometric functions, such as sin, cos, sinh, cosh, tan. It was later extended to logarithmic, hyperbolic and other functions by Walther (1971). The CORDIC algorithm contains the circumference systems, linear systems, hyperbolic systems, three kinds of rotation system. In a variety of rotation system is divided into the rotation mode and vector mode. To calculate the value of the natural logarithm, it should be the operating mode of the vector model in the hyperbolic system.

Iterative rotations: The function $x^2 - y^2 = 1$ is a flat hyperbolic function, its parametric form is (1):

$$\begin{cases} x = \cosh(t) \\ y = \sinh(t) \end{cases}, t \in (-\infty, \infty) \tag{1}$$

And, the matrix equation form is (2):

$$\begin{bmatrix} \cosh(t_2 + t_1) & \sinh(t_2 + t_1) \\ \sinh(t_2 + t_1) & \cosh(t_2 + t_1) \end{bmatrix} = \begin{bmatrix} \cosh t_2 & \sinh t_2 \\ \sinh t_2 & \cosh t_2 \end{bmatrix} \begin{bmatrix} \cosh t_1 & \sinh t_1 \\ \sinh t_1 & \cosh t_1 \end{bmatrix} \tag{2}$$

Corresponding Author: Yuanyuan Shang, College of Information Engineering, Capital Normal University, Beijing 100048, China

From vector (x_1, y_1) to another vector (x_2, y_2) hyperbolic rotation defined as follows:

$$\begin{cases} x_2 = x_1\cosh(t) + y_1\sinh(t) = \\ \cosh(t)[x_1 + y_1\tanh(t)] \\ y_2 = x_1\sinh(t) + y_1\cosh(t) = \\ \cosh(t)[y_1 + x_1\tanh(t)] \end{cases} \quad (3)$$

Its iterative form:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \cosh t_i \begin{bmatrix} 1 & \tanh t_i \\ \tanh t_i & 1 \end{bmatrix} \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} \quad (4)$$

The basic idea of the CORDIC algorithm is that not directly use the tanh t operator, but to select a series of special parameter t_1 satisfy $\tanh t_i = 2^{-i}$ for hyperbolic rotation. So multiplied by the operator $\tanh t_i$ becomes multiplied by the operator 2^{-i}. That is a simple shift operation.

Then add direction control factor d_i:

$$\begin{cases} x_{i+1} = \cosh t_i\ (x_i + d_i 2^{-i} y_i) \\ y_{i+1} = \cosh t_i\ (y_i + d_i 2^{-i} x_i) \end{cases} \quad (5)$$

z_i is defined to keep track of the parameter that has been changed:

$$z_{i+1} = z_i - d_i t_i \quad (6)$$

Set the initial vector (x_1, y_1). After N iterations, becomes:

$$\begin{bmatrix} x_{N+1} \\ y_{N+1} \end{bmatrix} = \left(\prod_{i=1}^{N} \cosh t_i\right) \bullet \begin{bmatrix} 1 & d_N 2^{-N} \\ d_N 2^{-N} & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & d_1 2^{-1} \\ d_1 2^{-1} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (7)$$

Parallel pipelined CORDIC: The CORDIC rotator is normally operated in one of two modes. The first called rotation, rotates the unit vector and then gradually iteration vector endpoints along the hyperbolic convergence on the parameter t ((x, y) = cosh t, sinh t)) point. The second mode called vectoring, rotes the input vector (rcosht, rsinht) to the x axis and gradually converges to (r, 0). Of solving practical problems is performed through cooperation and competition among them.

Rotation mode:

$$z_1 = t \to z_{N+1} = 0$$

if $z_i<0$ $d_i = -1$, else +1

Vectoring mode:

$$z_1 = 0 \to z_{N+1} = -\sum_{i=1}^{N} d_i t_i \approx t = \tanh^{-1}\left(\frac{y}{x}\right)$$

if $y_i<0$ $d_i = +1$, else +1

In vectoring mode the rotation produces:

$$\begin{cases} x_{N+1} = K_N\sqrt{x_1^{\ 2} - y_1^{\ 2}} \\ y_{N+1} = 0 \\ z_{N+1} = z_1 + \tanh^{-1}\left(\frac{y_1}{x_1}\right) \end{cases} \quad (8)$$

When $z_1 = 0$, then:

$$z_{N+1} = \tanh^{-1}\left(\frac{y_1}{x_1}\right) = \frac{1}{2}\ln\left(\frac{1+\frac{y_1}{x_1}}{1-\frac{y_1}{x_1}}\right) = \quad (9)$$

$$\frac{1}{2}\ln\left(\frac{x_1 + y_1}{x_1 - y_1}\right) = \frac{1}{2}\ln t \quad (x_1 = t+1, y_1 = t-1)$$

Thus:

$$\ln t = 2z_{N+1} \quad (10)$$

$$\left|\tanh^{-1}\left(\frac{y_1}{x_1}\right)\right| \le \sum_{i=1}^{N} t_i = 1.118, \mathrm{N} \to \infty \quad (11)$$

$$\left|\frac{y_1}{x_1}\right|_{\max} \approx 0.807 \quad (12)$$

If the initial input $x_1 = t + 1$, $y_1 = t - 1$ satisfy the convergence conditions, the input range is very limited. So how to increase the effective range of the input data is an important problem. The calculation shows that $\tanh^{-1} 12 = 0.9999999999244972$, so [-12, 12] can make the domain of the function close to (-1, 1), the complete domain of the function of x. y_1/x_1 can also be infinitely close to 1.

By including additional iterations for negative indexes i can increase the data valid input range (Hu *et al.*, 1991):

For $i \le 0$

$$\begin{cases} x_{i+1} = x_i + d_i y_i (1 - 2^{i-2}) \\ y_{i+1} = y_i + d_i y_i (1 - 2^{i-2}) \\ z_{i+1} = z_i - d_i \tanh^{-1}(1 - 2^{i-2}) \end{cases} \quad (13)$$

For i>0

Table 1: The floating-point form of t_i and its hexadecimal form

n	Floating-point form	Hexadecimal form
-5	2.7706	3FF4
-4	2.4221	37E9
-3	2.0716	2FD1
-2	1.7170	27A2
-1	1.3540	1F41
0	0.9730	1675
1	0.5493	0CAE
2	0.2554	05E5
3	0.1257	02E7
4	0.0626	0172
5	0.0313	00B9
6	0.0156	005C
7	0.0078	002E
8	0.0039	0017
9	0.0020	000B
10	0.0010	0006
11	0.0005	0003
12	0.0002	0001
13	0.0001	0000
14	0.0001	0000
15	0.0000	0000
16	0.0000	0000

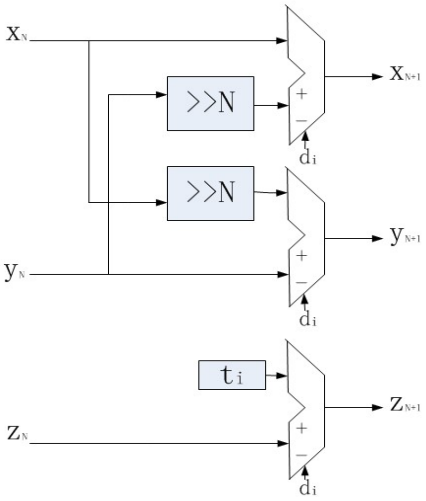

Fig. 1: A basic unit in the pipelined structure

$$\begin{cases} x_{i+1} = x_i + d_i y_i (1 - 2^{i-2}) \\ y_{i+1} = y_i + d_i y_i (1 - 2^{i-2}) \\ z_{i+1} = z_i - d_i \tanh^{-1}(1 - 2^{i-2}) \end{cases} \quad (14)$$

$$t_{\max} = \sum_{i=w}^{0} \tanh^{-1}(1 - 2^{-i}) + \tanh^{-1} 2^{-i} \quad (15)$$

When w = -5, it can satisfy the requirements of the input range, so we can start iteration from n = -5, the first formula of iteration (13) has been used when n is less than or equal to 0 by, otherwise the second formula of iteration (14) has been used. The amount of shift of each level of iteration unit is fixed, when the input data is 16 bits, 22 iterations structure can be used, the sequences of iterative shift are (7, 6, 5, 4, 3, 2, 1, 2 .... 16). And t_i is the change parameter for each iterative equation. The floating-point form of t_i and its hexadecimal form have been showed in the following Table 1.

FPGA IMPLEMENTATION

Pipelined-CORDIC: In practical applications, the natural logarithmic function calculator should be based on the needs of the target in between running speed and resource consumption. The pipelined-FPGA architecture can achieve a high computational speed, thereby greatly increasing the efficiency of the system. In this structure, each of the shifter is determined depth. Parameter accumulator's value as a constant direct connection to the accumulator on, do not need storage space and reading time. Figure 1 shows a basic unit in the pipelined structure. The direction control factor d_i defines the sign of the adder-subtractors. And t_i is the change parameter for each iterative equation. Its value is defined from Table 1.

The inputs and outputs have the same width (16 bits). Extends the bit width can improve the accuracy of the computation. So the data extended to 33, the highest bit is the sign bit.

The end: z = 1/2ln (t). The results of left shift one bit, the interception of the high 16 as a final output. The maximum data bit is the sign bit. For the maximum inputting value 65535, the real result is 11.0903, corresponding to the output of FPGA is FFFF. The remaining output divided by the corresponding value to get the right results.

In this study, the natural logarithmic function calculator is implemented by the FPGA chip-EP2C35F672C8, which costs a total of 4085 logic elements of 33216. The actual f_{max} is 94.18 MHz and satisfy the design requirements.

Function simulation There is a figure of the functional simulation for the calculator. It reflects the logic and timing relationships of the input and output of the calculator.

In the Fig. 2 of this simulation report, clk is a clock signal, xi, yi, zi are the input signal and xo, yo and zo is the output signal. It can be seen from that, the first output data generated after 22 cycles. Then after the rising edge of each clock will generate an output. This reflects the superiority of the pipeline structure of processing speed.

ANALYSIS OF RESULTS

It can be seen from the Table 2 that the natural logarithm calculator has high accuracy for five

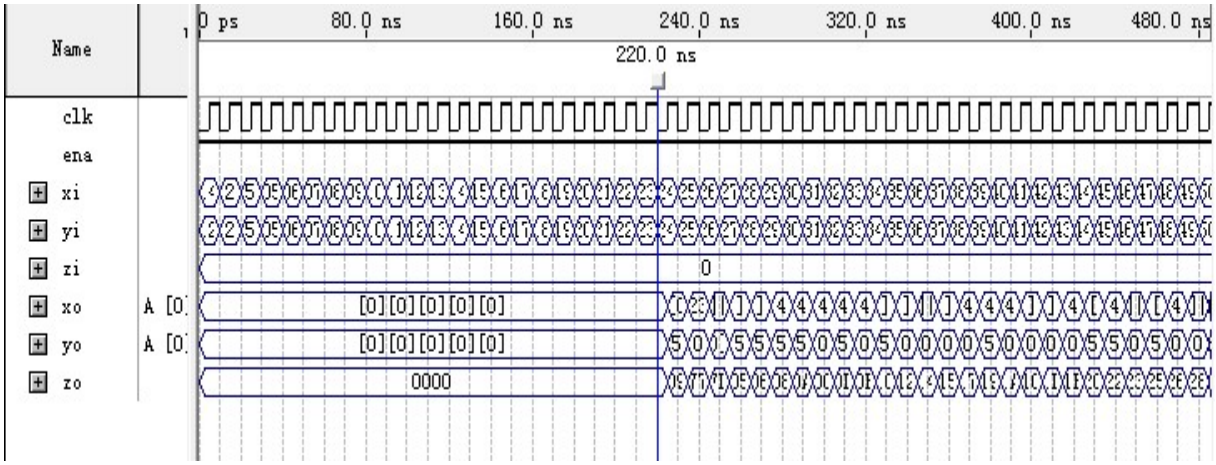

Fig. 2: Simulation report

Table 2: Result analysis

t	Output	CORDIC ln (t)	Ideal ln (t)	Relative error
13	3B32	2.5645	2.5649	-4e-4
255	7FEA	5.5415	5.5413	2e-4
1023	9FFE	6.9308	6.9305	3e-4
12001	D8D2	9.3931	9.3927	4e-4
64555	FFA6	11.0752	11.0753	-1e-4

randomly selected input values. The relative error values are below 10^{-4}, which can satisfy the accuracy requirements.

CONCLUSION

FPGA has a regular internal logic array and connection resources, suitable for digital signal processing tasks. Relative to the general-purpose chip serial arithmetic, FPGA has better parallelism and scalability. CORDIC is generally well-suited for handheld calculators, an application for which cost is much more important than speed. In this study, the CORDIC algorithm principle can be applied to the fast and accurate computation of the natural logarithm, given CORDIC algorithm can be completed by shift and addition operations in the FPGA. The pipelined-FPGA architecture can achieve a high computational speed. The relative error values are below 10^{-4}, it satisfies the accuracy requirements. Compared to other hardware realization method, this calculator satisfies the performance requirements in the aspects of operation speed, the calculation accuracy and resource consumption.

ACKNOWLEDGMENT

The authors wish to thank the helpful comments and suggestions from the teachers and colleagues in No. 108 lab of College of information Engineering, Capital Normal University.

REFERENCES

Guannan, C., Y. Kuntao, C. Rong and X. Zhiming, 2008. A gray-natural logarithm ratio bilateral filtering method for image processing. Chin. Opt. Lett., 6(9): 648-650-3.

Hu, X., R. Huber and S. Bass, 1991. Expanding the range of convergence of the CORDIC algorithm. IEEE T. Comput., 40: 13-21.

Michel, J. and P. Jean-Charles, 1995. Image dynamic range enhancement and stabilization in the context of the logarithmic image processing model. Signal Process., 41(2): 225-237.

Volder, J.E., 1959. The CORDIC trigonometric computing technique. IRE T. Electron. Comput., EC-8(3): 330-334.

Walther, J.S., 1971. A unified algorithm for elementary functions. Proceeding of Spring Joint Computer Conference, pp: 379-385.