Modified CORDIC Algorithm and Its Implementation Based on FPGA

Huan Li [1,2]
1.School of Information Science and Engineering
Shenyang Ligong University
Shenyang, China
e-mail: lihuan9999@yeah.net

Yan Xin [1]
2.School of Automatic Control
Nanjing University of Science and Technologe
Nanjing,China
e-mail: xinyan_lhy@163.com

***Abstract*—In conventional CORDIC algorithm,multiplier and a lookup table are needed to achieve calculation of multiple transcendental function,which will lead to hardware circuit complexity and lower operation speed. Aim at overcoming the shortcomings of traditional CORDIC algorithm, a modified CORDIC algorithm is proposed and implemented by FPGA program.The method does not need the module of correction factor and the lookup table,and just needs a simple shift and add-subtract to achieve the calculation of multiple transcendental function.So it can reduce hardware costs and improve operational performance.**

Keywords-CORDIC;stream-lined construction; modified algorithm; FPGA

I. INTRODUCTION

In the FPGA design,some special functions often need to be implemented,such as using FPGA to achieve some digital signal processing algorithms.If the algorithm uses a non-common (beyond) algebraic functions, we can use Taylor series to approximate this function, then the problem is simplified into a series of multiplication and addition operations, but the program is complex and the consumption of resources is so huge that it is almost not feasible[1]. A more effective method is based on the coordinate rotation digital computation(Coordinate Rotational Digital Computing， CORDIC)[2].CORDIC is presented by the J.D.Volder in 1959 and first used in navigation systems, making the vector rotation and the directional computing not need to do checking list of trigonometric functions,multiplication,square root and inverse trigonometric functions and other complex operations[3]. The basic idea of the algorithm is through a series of the continued deflection angle,fixed and associated with base number of calculate,to approximate the required rotation.Because of its basic operation unit being only shift and add and subtract, which laid a good foundation for the CORDIC algorithm combines with the rapid development of ultra large scale integrated circuit(Very Large Scale Integrated circuits,VLSI)technology.A significant reduction in FPGA resources makes these algorithms easier to be achieved in hardware, and thus meet the requirements of designer[4-5]. People pay more attention to its advantages, and it is widely used in computing real-time signal processing of high quality requirements, image processing and so on.

In this article,the FPGA implementation for CORDIC algorithm is discussed,and taking use of EDA tools and hardware description language Verilog HDL, the algorithm is implemented and verified.

II. CORDIC ALGORITHM PRINCIPLE

CORDIC algorithm is very suitable for FPGA implement, whoes main idea is to realize vector rotation through the iterative.In the constant iteration, a certain vector converge to the required objective. The basic principle of the algorithm shown in Fig.1.

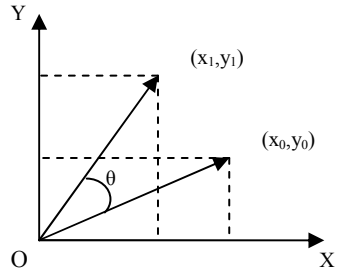

Figure 1. CORDIC algorithm schematic

It can be seen that if the vector (x_0, y_0) rotates angle θ, a new vector (x_1, y_1) can be got.

$$\begin{cases} x_1 = x_0 \cos\theta - y_0 \sin\theta \\ y_1 = y_0 \cos\theta + x_0 \sin\theta \end{cases} \tag{1}$$

Eq.(1) can be rearranged as (2).

$$\begin{cases} x_1 = \cos\theta[x_0 - y_0 \tan\theta] \\ y_1 = \cos\theta[y_0 + x_0 \tan\theta] \end{cases} \tag{2}$$

For easy hardware implementation, we set n times of rotation and the rotation angle of every times is θ_i and θ_i meeting $\tan\theta_i=2^{-i}$,so $\cos\theta_i = \sqrt{\frac{1}{1+2^{-2i}}}$, I-th rotation is expressed as (3).

978-0-7695-4249-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ICINIS.2010.30

$$\begin{cases} x_{i+1} = (x_i - \delta_i y_i 2^{-i})\sqrt{\dfrac{1}{1+2^{-2i}}} \\ y_{i+1} = (y_i + \delta_i x_i 2^{-i})\sqrt{\dfrac{1}{1+2^{-2i}}} \\ z_{i+1} = z_i - \delta_i \tan^{-1} 2^{-i} \end{cases} \quad (3)$$

Where,after I-th rotation,angle changes is z_i.The direction of per rotation is δ_i, which is equal to the sign of z_i,that is δ_i=sign(z_i).When δ_i=+1, rotates counterclockwise, and when δ_i=-1, rotates clockwise. $\sqrt{\dfrac{1}{1+2^{-2i}}}$ is the correction factor for each level, that is, the rotation vector' module long changes of each level.For a certain length, the total correction factor is a constant. If the total series of rotation is N, the total correction factor is expressed with the K as (4).

$$K = \prod_{i=0}^{N-1} \sqrt{\frac{1}{1+2^{-2i}}} \quad (4)$$

Take 16-bit as example, K = 0.607252935.

We can first correct data for the operation, so that each level of operations can be reduced to (5).

$$\begin{cases} x_{i+1} = (x_i - \delta_i y_i 2^{-i}) \\ y_{i+1} = (y_i + \delta_i x_i 2^{-i}) \\ z_{i+1} = z_i - \delta_i \tan^{-1} 2^{-i} \end{cases} \quad (5)$$

From (5), it can be seen that all the operations are simplified to add,subtract and shift operations.When given the initial input data is x_0=K and y_0=0, z_0=θ.After n times of iteration, the results are as follows:

$$\begin{cases} x_n = \cos\theta \\ y_n = \sin\theta \\ z_n \to 0 \end{cases} \quad (6)$$

By the analysis of formula (6),we can see that the rotation mode of CORDIC algorithm in circle system can be used to calculate the input angle of the sine, cosine and so on.

III. THE IMPLEMENTATION METHODS OF CORDIC ALGORITHM

The implementation methods CORDIC algorithm are in two ways,a simple state machine method and high-speed stream–lined processor.The former mainly adopts the iterative approach, the latter uses pipelining. Pipeline structure CORDIC is highlighted here.

Although occupied hardware resources more by stream-lined CORDIC, but its structure can improve data throughput. From the present trend of VLSI,chip gate resources are relative affluence,which constraints less on the implementation scale of stream-lined CORDIC. In addition, the stream-lined CORDIC does not exist an iterative feedback loop so as to make the unit structure more rules and conducive to VLSI implementation.Figure 2 gives the general pipeline structure of CORDIC algorithm.

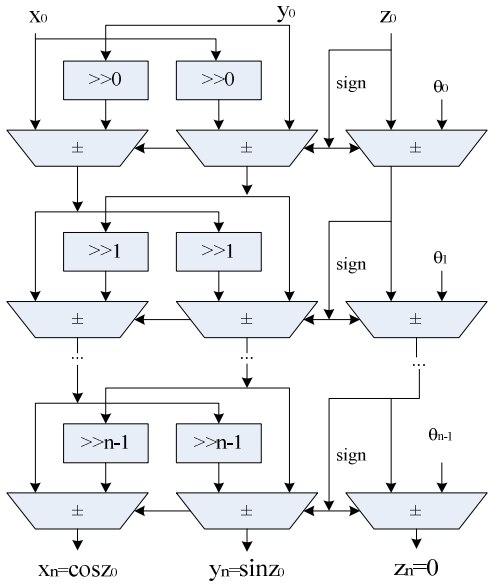

Figure 2. Pipeline structure flow chart of CORDIC algorithm

IV. IMPROVEMENT OF CORDIC ALGORITHM

It can be seen from (2) that the traditional CORDIC vector rotation algorithm is not perfect because the process of rotation will change the module length of vectors, which need to use module correction factor.This will bring the trouble of multiplication and the lookup table to every step of iteration, thus increase the difficulty of hardware implementation and reduce the data processing speed.

Taking into account the iterative sequence can cover the angle range: $\sum_{i=0}^{n-1} \tan^{-1} 2^{-i}$, if the direct use of n-level iterative sequence: n=0,1,…， n-1, then the cover angle range is -99.9º～+99.9º.But the general requirements for coverage is [-π， π],it's needed to correct for iteration. Therefore, the need to increase a specific initial step before the first iteration to expand the angle coverage, that is,under the positive and negative of input phase, vector is firstly rotated clockwise or counterclockwise 90º so as to achieve coverage requirements. The mathematical expression of this step is in (7).

$$\begin{cases} x_1 = x_0 \cos 90° + y_0 \sin\delta 90° = \delta y_0 \\ y_1 = y_0 \cos 90° - x_0 \sin\delta 90° = \delta x_0 \\ z_1 = z_0 - \delta 90° \end{cases} \quad (7)$$

Among which δ=sign(z_0).

CORDIC algorithm is used to replace the lookup table to save a lot of RAM resources, but also has brought more LE consumption, which needs to be considered to reduce in the design.

For small angles sine and cosine values, there is:

$$\begin{cases}\sin\theta \approx \theta \\ \cos\theta \approx 1\end{cases}, \text{ when } \theta \to 0 \tag{8}$$

In CORDIC algorithm's iterative computations of finite precision, a certaint progression of coordinate rotation angle is also a small angle close to 0. Using the feature, we can improve on the CORDIC algorithm. The following is the describes about the improvement of CORDIC algorithm taken output width of 16-bit as an example.

Notes that after 9 iterations, the remaining angle is: θ=0.003906, there is (9).

$$\begin{cases}\sin\theta = 0.00390599 \approx \theta \\ \cos\theta = 0.99999237 \approx 1\end{cases} \tag{9}$$

Let $z_0 = z_8 + \sum_{i=0}^{8} \delta_i \tan^{-1} 2^{-i}$, z_8 is the remaining angle after 9 times of iterations. We can use the conventional CORDIC algorithm for the first 9 times of iterations, and take directly the rotation changing formula of initial angle for the next levels:

$$\begin{cases}x_9 = x_8 \cos z_8 - y_8 \sin z_8 \\ y_9 = y_8 \cos z_8 + x_8 \sin z_8\end{cases} \tag{10}$$

We have known $z_8<2^{-8}$ and for the 16-bit output accuracy, $\cos z_8=1$, $\sin z_8=z_8$, then (10) can be written as (11)

$$\begin{cases}x_9 = x_8 - y_8 z_8 \\ y_9 = y_8 + x_8 z_8\end{cases} \tag{11}$$

From (11) we can know that, for the CORDIC operation of 16-bit output width, only nine times of iterations and one level of initial angle rotation operations is needed. This structure can effectively improve the efficiency of CORDIC operations and save resources required largely.

The multiplication in (11) can be calculated by the parallel addition, that will change the multi-level cascading addition operation into a one level synthesize carry storage adder. Synthesize carry storage adder's expression is (12).

$$\begin{cases}x_9 = x_8 + \delta_8 y_8 \sum_{i=9}^{16} \alpha_i 2^{-i} \\ y_9 = y_8 + \delta_8 x_8 \sum_{i=9}^{16} \alpha_i 2^{-i}\end{cases} \tag{12}$$

When $\delta_8=1$, α_i is the i-th bit of z_8.When $\delta_8=-1$, α_i is the i-th bit of z_8's binary counter code. The structure of carry storage adder is shown in figure 3.

Adding the pre-iteration, the traditional CORDIC algorithm needs 17 level pipeline to achieve16-bit output width. Only 9 level of pipeline and one level of carry adder storage is needed in modified CORDIC algorithm.The overall structure of improved CORDIC algorithm is shown in Figure 4.

When this pipeline structure normally working, after initial delay, each time of a cycle accomplishment will generate a new output value, that is, only need one clock cycle to finish a data output. For heightening the precision, we can increase the output bit-width and the pipeline levels correspondingly.

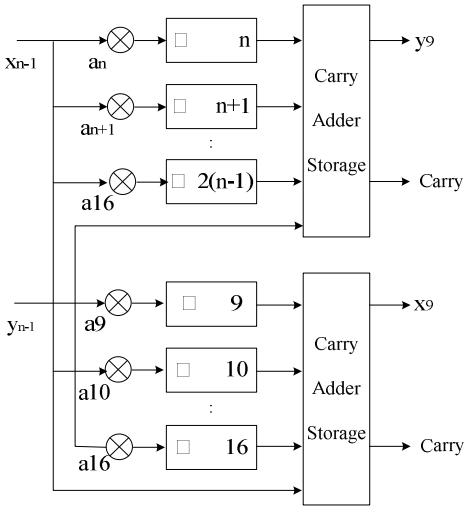

Figure 3. The structure of carry storage adder

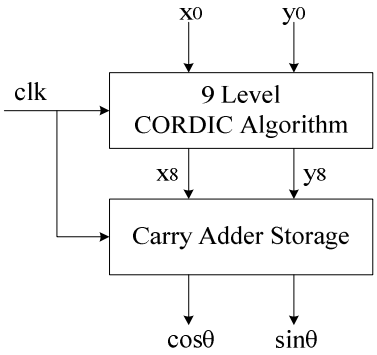

Figure 4. The overall block diagram of improved CORDIC algorithm

V. THE FPGA IMPLEMENTATION FOR CORDIC ALGORITHM

Using In the design, the FPGA chip selected is EP2C8T144C8 of Cyclone II device family developed by Altera Corporation[6]. After the RTL-level coding to complete the circuit describing by Verilog HDL language, compilation and simulation is implemented in the Quartus II 9.1 software platform,and Modelsim SE 6.4 is used for joint simulation. The simulation results is shown in figure 5.

Table 1 shows the simulation results and theoretical values. Comparison shows that simulation results are basically consistent with the theoretical values.The mean square error of simulation value is 0.001 in the modified CORDIC algorithm.Comparing with the traditional algorithm,the mean square error has reduced by 60%.So the

iterative operation by CORDIC algorithm can greatly improve the accuracy.

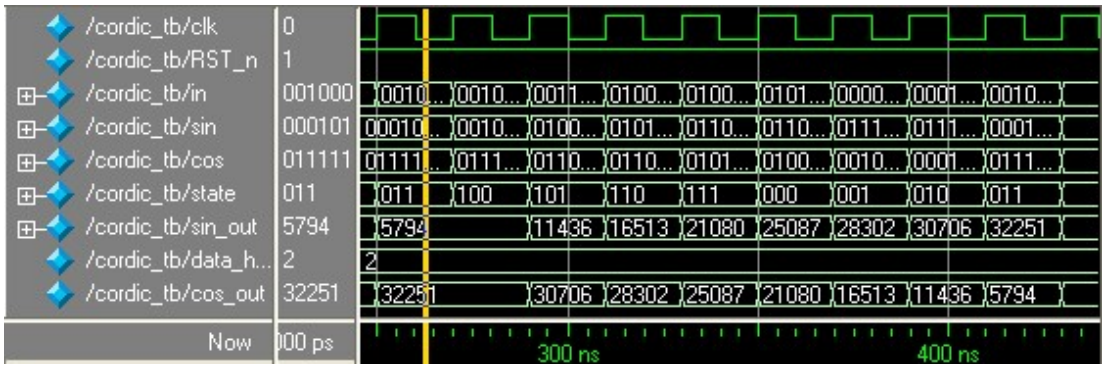

Figure 5. The FPGA simulation results

Table 1 The comparison of some typical simulation output phase of the sine and the theoretical ones

input	17 bit the fixed-point arc (Decimal)	Converted to Hex	Theoretical sine value	Simulation value	Converted to decimal
10	5719.094893	01657	0.173648178	5794	0.176819
20	11438.18979	02CAE	0.342020143	11436	0.348999
30	17157.28468	04305	0.5	16513	0.503937
40	22876.37957	0595C	0.64278761	21080	0.643311
50	28595.47446	06FB3	0.766044443	25087	0.765594
60	34314.56936	0860A	0.866025404	28302	0.863708
70	40033.66425	09C61	0.939692621	30706	0.937073
80	45752.75914	0B2B8	0.984807753	32251	0.984222

VI. CONCLUSION

The compilation and simulation by Quartus II has verified the correctness of the Verilog HDL program of CORDIC algorithm.The simulation results and theoretical values are consistent.The accuracy of the method can be close to 10^{-6}. The algorithm can meet the hardware requirements for modular and regularization.It can give full play to the advantages of hardware and usage of hardware resources to achieve an optimization plan for combination of hardware and algorithms.

REFERENCES

[1] Uwe Meyer Baese,Digital signal processing with field programmable gate arrays,Beijing:Tsinghua University Press, 2003,pp.57-89.

[2] Hu Y H,CORDIC-based VLSI architecture for digital signal processing,IEEE SP Mag,2002(7),pp.17-35.

[3] Li Yan,"CORDIC algorithm application in the DSP algorithm hardware implement,"Modern electronic technology, 2002(6),pp.85-89.

[4] Xu Guang-hui,Embedded development and application base on FPGA,Beijing: Electronic Industry Press, 2006,pp.54-66.

[5] Li Hong-wei and Ruan Si-hua,Base on Quartus II's FPGA/CPLD design,Beijing: Electronic Industry Press, 2006.

[6] Wang Cheng and Wu Ji-hua,Altera FPGA/CPLD design,Beijing: People Posts & Telecom Press, 2005.