International Journal of Hybrid Information Technology
Vol.8, No.6 (2015), pp.217-228
http://dx.doi.org/10.14257/ijhit.2015.8.6.21

An Optimization of CORDIC Algorithm and FPGA Implementation

Rui Xu[a], Zhanpeng Jiang[b], Hai Huang[c], Changchun Dong[d]

Department of Integrated circuits design and integrated systems,Harbin University of Science and Technology ,Harbin, Heilongjiang, China
[a]ruirui_210@hrbust.edu.cn, [b] zhanpeng_jiang@126.com,
[c]Huanghai@hrbust.edu.cn [d]dongchangchun@126.com

Abstract

ASIC and FPGA ASIC and FPGA are considered to be the ideal platform for special fast calculations because of the hardware structure, and how to achieve computational algorithm by is the hotpot of research. The CORDIC (Coordinate Rotational Digital Computer) can break the basis functions down to operations of shift and addition or subtraction, which can be used to lay the foundation for the realization of complex logic. But the functions selected by traditional CODIC for angle encoding are too complex, which will lead to some problems, such as too much of area consumption and large delay. In this paper, an optimization of CORDIC algorithm are proposed, which reduce the consumption of Adders and comparators, decrease the complexity and delay of the algorithm implement in hardware. The proposed algorithms are modeled in Verilog Hardware Description Language and implemented with FPGA. The simulation results show that the functions of sine and cosine are realized successfully, and the proposed algorithm not only improves the computation speed but also reduces the system hardware resources.

Keywords: *CORDIC, FPGA, ultra-high-speed integrated circuit, optimization*

1. Introduction

CORDIC algorithm is the best choice to achieve the functions of transcendental functions such as trigonometric, inverse trigonometric, exponential function , logarithmic function since that the CORDIC algorithm is provided with a simple structure, and characteristic of saving resources and high efficiency, as while as CORDIC algorithm is used widely for matrix operations, especially possess significance for development of transplant of highly complex operations in FPGA, such as analysis algorithms for multidimensional data array[1]. CORDIC algorithm is an iteration algorithm and commonly used to calculate basic arithmetic functions. The principle of the CORDIC is to use the deflection of the angles associated with the cardinal number, rather than get the desired angle. Thus, the principle can be considered as a numerical approximation methods of calculation. Because the fixed angle is related with cardinal number, the operations of computation include only shift and addition or subtraction. Therefore, it will cost less FPGA (Field Programmable Gate Array) resources than conventional calculation methods, such as multiplication and division. The conventional calculation methods are difficult to achieve or can not meet the designer's requirements. The appearance of CORDIC algorithm is to solve this problem, the CORDIC can greatly saving FPGA resources to get better implement in hardware, which can achieve the requirements of the designer.

In this paper we discuss how to implement the CORDIC algorithm with FPGA, model it in Verilog hardware description language, simulate it by EDA tools, and analyze the performance from aspects of the resource consumption, delay, the number of iteration and the accuracy of computation.

2. Principle of CORDIC Algorithm

CORDIC algorithm is mainly used to solve the problem in two-dimensional vector rotation plane. It replaces the rotation operations with simple ones, such as shift and addition or subtraction. Through the algorithm we can divide the larger target rotation angle θ into several consecutive smaller deflection angles α_i, so as to achieve the process of rotation. The delay is mainly determined by the number of iterations and the time spent in each iteration stage. Since the original CORDIC algorithm is designed without the mechanism which could terminate iterations, so a fixed number of iterations must be required. There will be redundancy iteration in this process, which will make a large delay. To solve this problem, many researchers have proposed different methods [2].

CORDIC algorithm has three different rotation systems, circumferential systems, linear systems and hyperbolic systems [3]. In this paper, the rotation process of CORDIC algorithm is only derived by circumferential system, and its schematic is shown in Fig. 1.

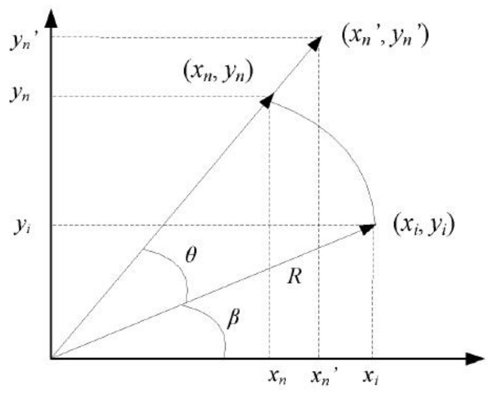

Figure 1. Schematic of CORDIC Algorithm Rotation

We will obtain a new vector (x_n, y_n) by rotation of the initial vector (x_i, y_i), the coordinates of the new vector can be expressed as:

$$x_n = \sqrt{(x^2+y^2)}\cos(\theta+\beta) = \cos(\theta)x_i - \sin(\theta)y_i$$
$$y_n = \sqrt{(x^2+y^2)}\sin(\theta+\beta) = \sin(\theta)x_i + \cos(\theta)y_i \tag{1}$$

The (1) can be written in matrix form as:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_i \\ y_u \end{bmatrix} \tag{2}$$

Assuming $\tan\alpha i=2^{-i}$, the angle of rotation of each step $\alpha i=\delta i$ arctan2^{-i} can be approximated as:

$$\theta = \sum_{i=0}^{N} \sigma_i \alpha_i + \varepsilon = \sum_{i=0}^{N} \sigma_i \arctan 2^{-i} + \varepsilon \tag{3}$$

In (3), $\delta i=\{1, -1\}$, the sign of δ_i determines the direction of rotation, which would be close to the target vector if δ_i =1, else would be close to the opposite direction if δ_i =-1. Then the remaining angle after each rotation can be expressed as: $Z_{i+1}= Z_i - \delta_i\alpha_i$, where δ_i=sign(Z_i).

Assuming that we could complete the rotation angle θ by N times of iterations, then the rotation process may be represented by (5).

$$\begin{bmatrix} x_{N+1} \\ y_{N+1} \end{bmatrix} = \prod_{i=1}^{N} \begin{bmatrix} \cos(\alpha_i) & -\sin(\alpha_i) \\ \sin(\alpha_i) & \cos(\alpha_i) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$
$$= \prod_{i=1}^{N} \cos(\alpha_i) \prod_{i=1}^{N} \begin{bmatrix} 1 & -\tan(\alpha_i) \\ \tan(\alpha_i) & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{5}$$
$$= \prod_{i=1}^{N} \frac{1}{\sqrt{1+2^{-2i}}} \prod_{i=1}^{N} \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Where $k_i = 1/\sqrt{1+2^{-2i}}$, k_i would converge to a constant as while as the increasing of the number of iterations. The gain constant K of the rotational operation can be expressed as:

$$K = \prod_{i=0}^{N} k_i = \prod_{i=0}^{N} \frac{1}{\sqrt{1+2^{-2i}}} \tag{6}$$

The value of K depends on the number of iterations N, and K was usually called focus constant or scaling factor. Generally, we could calculate the scaling factor and the rotation operation separately. The rotation operation was carried out according to (7).

$$\begin{bmatrix} x_{N+1}' \\ y_{N+1}' \end{bmatrix} = \prod_{i=1}^{N} \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{7}$$

However, the range of δ_i would be extended to {-1, 0, 1}, so as to skip the unnecessary rotation and greatly reduce the number of iterations. The gain K would be no longer a constant, therefore, after each rotation step we would not only update coordinates of xi and yi but also the scaling factor ki for each step [4].J. S. Walther et al. proposed a CORDIC algorithm in 1971, which unified rotation of CORDIC under the circumference coordinate system, hyperbolic system and linear systems into a single iteration equation, which was shown as:

$$\begin{aligned} &(\forall i = 0,1,\ldots,\ N) \\ &X_{i+1} = k_i * \left(X_i - m\sigma_i 2^{-s(mj)} Y_i\right); \\ &Y_{i+1} = k_i * \left(Y_i - \sigma_i 2^{-s(mj)} X_i\right); \\ &Z_{i+1} = Z_i - \sigma_i \alpha_{mj} \end{aligned} \tag{8}$$

Where m={1, 0, -1} for the peripheral system is a coordinate system , a linear system or a hyperbolic systems respectively. The αmj is a constant angle, could be unified expressed as:

$$\alpha_{mj} = \frac{1}{\sqrt{m}} \tan^{-1}\left(\sqrt{m} * 2^{-s(mj)}\right) \tag{9}$$

Where, S(mj) is the shift sequence that could be represented by (10) in different systems.

$$\begin{aligned} S(mj) &= 0,1,2,3,4,5,\ldots,\ N. \quad (m=1) \\ &= 0,1,2,3,4,5,\ldots,\ N. \quad (m=0) \\ &= 0,1,2,3,4,5,\ldots,\ N. \quad (m=-1) repeat\left(3^{i+2}-1\right)/2 \end{aligned} \tag{10}$$

Uniform scaling factor can be expressed as:

$$\begin{aligned} k_i &= 1/\sqrt{1+m\sigma_i^2 * 2^{-S(mj)}} \\ K &= \prod_{i=0}^{N} k_i = \prod_{i=0}^{N} 1/\sqrt{1+m\sigma_i^2 * 2^{-S(mj)}} \end{aligned} \tag{11}$$

The unity of iteration expression under different systems laid the theoretical foundation for the same hardware to implement multiple functions, the results in different systems and modes when the entry was (x_0, y_0, z_0) would be shown as in Table 1.

Table 1. Results of CORDIC Algorithm in Different Modes

CORDIC	Mode of rotation	Mode of vector
Circumference System (m=1)	$x_n = \frac{1}{K_c}(x_0 \cos z_0 - y_0 \sin z_0)$ $y_n = \frac{1}{K_c}(y_0 \cos z_0 + x_0 \sin z_0)$	$x_n = \frac{1}{K_c}\sqrt{x^2+y^2}$ $z_n = z_0 + \arctan(y_0 / x_0)$

CORDIC	**Mode of rotation**	**Mode of vector**
Linear Systems（m=0）	$x_n = x_0$ $z_n = z_0 + y_0 * x_0$	$x_n = x_0$ $z_n = z_0 + y_0 / x_0$
Hyperbolic systems（m=-1）	$x_n = \frac{1}{K_h}(x_0 \cosh z_0 - y_0 \sinh z_0)$ $y_n = \frac{1}{K_h}(y_0 \cosh z_0 + x_0 \sinh z_0)$	$x_n = \frac{1}{K_c}\sqrt{x^2 - y^2}$ $z_n = z_0 + \arctan(y_0 / x_0)$

The CORDIC algorithm based on angle encoding was same as the traditional ones, it assembled linearly rotation angle θ by a series combination of a small angle, however, the difference was that the rotational direction of the vector could be zero, i.e., $\alpha_i = \{-1, 0, 1\}$. The use of greedy mechanism, making each selection from the remainder of the rotation angle is the angle nearest. The pseudo-code of angle encoder CORDIC algorithm is shown as:

```
initial ： $\theta(0)=\theta,\{\sigma(i)=0;0\le i\le n-1\},\ k=0.$
repeat until ： $|\theta(k)|<\alpha(n-1)$
then select $i_k, 0\le i\le n-1$， $||\theta(k)|-\alpha(i_k)| = \min_{0\le i\le n-1}||\theta(k)|-\alpha(i_k)|$
$\theta(k+1)=\theta(k)-\sigma(i_k)\alpha(i_k)$，  $\sigma(i_k)=sign(\theta(k))$
```

To expand the scope of the convergence to the range of (0, π/2), the algorithms took advantage of interval folding technique, which under the following rules: if the range of θ is θ>2π, let θ=θ-2π; if the range is from π to 2π, replacing the $[x\ y]_T$ with $[-x\ -y]_T$, and let θ=θ-π; if the range is π>θ>π/2, replacing the $[x\ y]_T$ with $[-x\ -y]_T$, and let θ=θ-π/2.

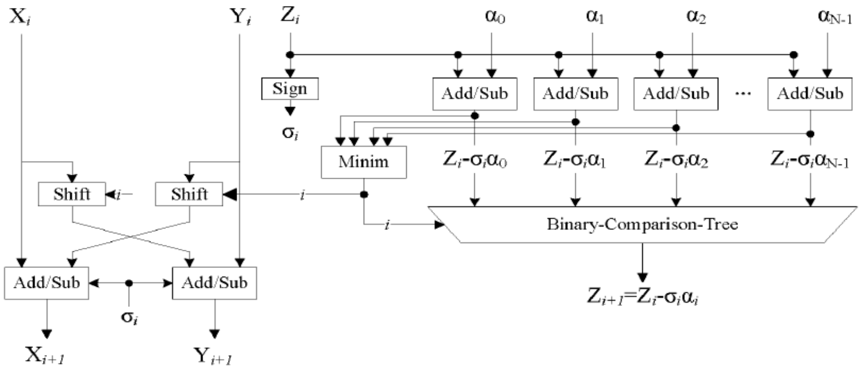

Figure 2. Implementation of Angle Encoder CORDIC

Since the rotation angles of the CORDIC are known, the number of iterations is able to reduce using angle encoder method at least 50% with N-bit precision. The implementation of angle encoder CORDIC is shown in Figure 2. In Figure 2, the algorithm requires an N-bit adder /subtracter, and a comparison unit for getting the minimum. Since all of these operations are on the critical path of the each iteration, the time of iteration and consumption of area are greatly increased. Angle encoder CORDIC algorithm could greatly reduce the number of iterations at the cost of increasing the latency of a single iteration, and skip some unnecessary rotation angle so that the scaling factor is no longer a constant [6].

3. Problems and Optimization of CORDIC Algorithm

In this section, we will analysis limitations of CORDIC algorithm from the research object of rotation angle θ_i, and put forward the corresponding optimization measures to solve these problems [6].

3.1 Relationship between the Number of Iterations and Accuracy of Data

Thinking that the angle of rotation Z could be negative in the iterative process, we represent Z with b-bit two's complement, and the minimum of angle is 2^{-b}. To make sure that the last rotation is meaningful, the angle of it must be less than the minimum value. Since all the angles of rotation are stored in tables of ROM, the minimum angle stored in the table must be less than 2-b. Assuming that the angle stored in ROM being also represent with a-bit two's complement, the minimum angle, 2^{-a}, should meet the requirement of $a \geq b$. The sequence of angle stored in ROM table is $\arctan(2^{-i+1})$, i=0,1,2….N-1, number N is the word length of the CORDIC algorithm computation. When number i is large enough, $\arctan(2^{-i+1}) \approx 2^{-i+1}$, $\arctan(2^{-N+1}) \leq 2^{-N+1} \leq 2^{-a}$, $N \geq a-1$.Thus, at least 15-level pipelines had reached the desired for meet the accuracy of the CORDIC algorithm with 16-bit word length of computation.

3.2 Limitations of CORDIC Algorithm

For the N-level pipeline, the variable θ is the rotation angle, and the range of number i is from 0 to N-1, θi= arccan (2^{-i}). The values of θi as the entry of every level pipeline, should be calculated in advance, and stored in table of ROM.

The sequences of 16-bit θ_i taken part in symbolic computation are represented with two's complement in Table 2. In Table 2, we can see that as the number of pipeline level increases, the capacity of the ROM table grows exponentially, and the area of the system will also increase.

Table 2. Datum in ROM Table

θ_i	Tangent	Radians	Binary
1	arctan（2^0）	0.7853981634	0110010010000111
2	arctan（2^{-1}）	0.4636476090	0011101101011000
3	arctan（2^{-2}）	0.2449786631	0001111101011011
4	arctan（2^{-3}）	0.1243549945	0000111111101010
5	arctan（2^{-4}）	0.0624188100	0000011111111101
6	arctan（2^{-5}）	0.0312398334	0000001111111111
7	arctan（2^{-6}）	0.0156237286	0000000111111111
8	arctan（2^{-7}）	0.0078123411	0000000011111111
9	arctan（2^{-8}）	0.0039062301	0000000001111111
10	arctan（2^{-9}）	0.0019531225	0000000000111111
11	arctan（2^{-10}）	0.0009765622	0000000000011111
12	arctan（2^{-11}）	0.0004882812	0000000000001111
13	arctan（2^{-12}）	0.0002441406	0000000000000111
14	arctan（2^{-13}）	0.0001220703	0000000000000011
15	arctan（2^{-14}）	0.0000610352	0000000000000001

As can be seen in Fig. 2, it requires multiple iterations for once computation of CORDIC algorithm and the direction of iteration must be determined by the result of the last iteration. The more the number of iteration is, the more the number of direction determination must be required, which will undermine the speed of operation.

3.3 Range of Angle

Assuming the number of iteration is N, the range of rotation is shown as:

$$-\sum_{n=0}^{N-1}\arctan(2^{-n}) \le \theta \le \sum_{n=0}^{N-1}\arctan(2^{-n}) \tag{13}$$

The range of angle corresponding with N would be calculated according to the formula given. In Table 3, the maximum range of rotation angle is -99.88° ≤ θ ≤ 99.88°, which could not achieve a complete cycle. To make sure that CORDIC algorithm is convergence, the sum of rotation angles must be bigger than the angle of rotation which is actually required, and the input angle should must be pretreated [7].

Table 3. Range of Angle of Rotation Corresponds to N

N	θmax	N	θmax
1	45°	8	99.44°
2	71.56°	9	99.67°
3	85.60°	10	99.77°
4	92.73°	11	98.83°
5	96.30°	12	99.85°
6	98.09° 98.99°	13	99.87°
7	98.99°	≥ 14	99.88°

3.4 Optimization of CORDIC Algorithm

Since computation of CORDIC algorithm needs calculating the arc tangent function, the usual process is to calculate the corresponding datum shown in Table. 2 previously then store them in the ROM, which took too much hardware resources and make the computation speed slow. If we reduce the number of iterations without affecting the computation accuracy, we can effectively improve the speed of operation. The proposes a feasible method by studying the rotation angle θ_i [8][9].

By taking use of the Taylor series of arc tangent function, we could obtain the following equations (14).

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} ....,|x| \le 1 \tag{14}$$

We could expand $\theta_i = \arctan\left(2^{-i}\right)$ to Taylor series by i=1, 2, 3…..N, and get

$$\theta_i = \arctan\left(2^{-i}\right) = 2^{-i} - \frac{1}{3}\cdot 2^{-3i} + \frac{1}{5}\cdot 2^{-5i} - \frac{1}{7}\cdot 2^{-7i}...... \tag{15}$$

Obviously, the difference between arctan(2-l) and 2-l decreases rapidly as while as the increasing of the number i.

If we replace arctan(2-i) with 2-l from the m-level iteration, there is no effect on the calculation accuracy of the final results. We could use shift operation instead of process of checking the ROM table, which speeding up the computation and reducing the occupancy of resources. The key of the theory is that the tolerance introducing by replacing arctan(2-i) with 2-l. For the computation of CORDIC algorithm, it is unnecessary to check arctan(2-i) from ROM table, if replacing it with 2-i when the number of iteration is greater than or equal to m, which reduces the time accessing to ROM, enhances the speed of computation, and reduces the ROM resource by two-thirds[10].

According to the following Taylor series (16), we could introduce variable l, when i ≥ l, the formula $|\cos\theta_{i-1}| \le 2\text{-}n$ must be established; when N = 15, the I = 8 can be obtained; when N = 31, the l = 16 can be determined[11].

$$\cos\theta_i = \frac{1}{\sqrt{1+2^{-2i}}} = 1 - 2^{-2i-1} + 3\cdot 2^{-4i-3} + ..., \quad i = 0,1,2,....., N-1 \tag{16}$$

$$\left|\cos\theta_i - 1\right| = \left|-2^{-2i-1} + 3\cdot 2^{-4i-3} + ...\right| \le 2^{-2i-1} \qquad (17)$$
$$i = 0,1,2,....., N-1$$

So, equation (7) can be simplified as (18), and the correction factor is 1which is used to simplify the operation .

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -\delta_i 2^{-i} \\ \delta_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad (18)$$

Similarly, if i ≥ 8, (8) can be simplified as (19),

$$\begin{aligned} x_{i+2} &= x_{i+1} - \delta_{i+1} 2^{-i-1} y_{i+1} = x_i - \left(\delta_i 2^{-i} + \delta_{i+1} 2^{-i-1}\right) y_i \\ y_{i+2} &= y_{i+1} - \delta_{i+1} 2^{-i-1} x_{i+1} = y_i - \left(\delta_i 2^{-i} + \delta_{i+1} 2^{-i-1}\right) x_i \\ z_{i+2} &= z_{i+1} - \delta_{i+1} \arctan(2^{-i-1}) = z_i - \delta_i 2^{-i} + \delta_{i+1} 2^{-i-1} \end{aligned} \qquad (19)$$

If we select δi and δ_{i+1} properly, two-step iteration would be combined into a single step. Thus, we could process two adjacent bits of rotation angle Zi in the one iteration, nature of which was actually replacing its corresponding bit with 0, tending the rotation angle to 0. It reduces the number of stages of the pipeline and improves speed of computation. The value of (δ_i,δ_{i+1}) was shown in Table 3.

Table 4. Value of (δ_i,δ_{i+1})

Value of (z_0,z_{i+1},z_{i+2})	**Value of δ_i, δ_{i+1}**
（0,1,1）	(1,0)
(0,0,1)or(0,1,0)	(0,1)
(0,0,0)or(1,1,1)	(0,0)
（1,0,0）	(-1,0)
(1,0,1)or(1,1,0)	(0,-1)

Based on the above analysis, the improved CORDIC algorithm works only when the value i = max (m, l) and N = 15). Therefore, it is necessary to use the combination of (7), (18) and (19) to calculate a value of function. For example, if N = 16, we firstly took nine times of iteration with formula (7), and corrects mode correction factor as $K = \prod_{i=0}^{8} \frac{1}{\sqrt{1+2^{-2i}}} \approx 0.607259$, then took three times of iteration for the following six stages, and corrects mode correction factor as one [8]. So the whole process needs twelve times of iteration, reduces three stages of pipeline and six storage units, cuts down the consumption of hardware unit, reduces the ROM accessing times, and reduces the computation time. Thus, under the premise of guaranteed system performance, the optimization of CORDIC algorithm economizes hardware resources and enhances the speed of computation [12-13].

3.5 Adjustment of Range of the Input Angle

As mentioned above, the range of the CORDIC rotation angle is from -99.88° to 99.88°, not cover the whole circumference, which limits the scope of the calculation of the algorithm. The solution usually used is to take multiple iterations with the number i=0 to make sure that the angle of CORDIC algorithm covering all the four quadrants.

However, the correction factor is not easy to determine in advance, and it is difficult to calculate it immediately. Another approach called sub-quadrant method is taken in this paper. This method takes advantage symmetry of trigonometric, and converts all of the perspective of the whole cycle into the first quadrant. Then it preserves the phase information according to the rules of transition, and puts it into the module of computation for CORDIC algorithm. After that, it restores to the original angle of the sine and cosine of the phase information.

Assuming the input angle was z0, the following four cases indicated the specific rules of conversion for 16-bit CORDIC algorithm, which is shown in Table 5.

Table 5. Angle Conversion Rule Table and Restore Function

Original angle z_0	Transformed angle z_0'	x_n	y_n
$0^o \le z_0 \le 90^o$	z_0	x_{15}	y_{15}
$90^o \le z_0 \le 180^o$	z_0-90^o	-y_{15}	x_{15}
$180^o \le z_0 \le 270^o$	z_0-180^o	-x_{15}	-y_{15}
$270^o \le z_0 \le 360^o$	z_0-270^o	y_{15}	-x_{15}

4. Implementation of the CORDIC Algorithm based on FPGA

FPGA is a semi-custom integrated circuit coming from ASIC (Application Specific Integrated Circuit, ASIC), which not only solve the defect of custom circuits, but also overcome the limitation of the number of the original gates of programmable devices.

4.1 Introduction of Tools for Development

We can complete most developments of digital devices by FPGA, such as CPU (Central Processing Unit), a circuit of 74 series. We can reduce design time and area of PCB (Printed Circuit Board) and improve system reliability by using FPGA to develop digital circuits. The process of FPGA design is shown in Fig. 3.

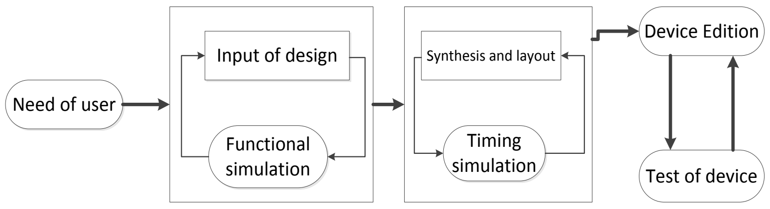

Figure 3. The Process of FPGA Design

System engineers could connect the internal logic blocks in FPGA together by editing of connection as needed, just like placing a test circuit board into a chip. The logic blocks and connection of a development of FPGA could be edited by designer, so to complete the required logic functions [10].

4.2 Design of the System

The framework of the optimized CORDIC algorithm based FPGA is shown in Fig. 4, which consisted of five modules, including an UART (Universal Asynchronous Receiver/Transmitter) controller, a cache allocator for the initial value, a pre-processing unit, an unit of optimized CORDIC and the post-processing unit.

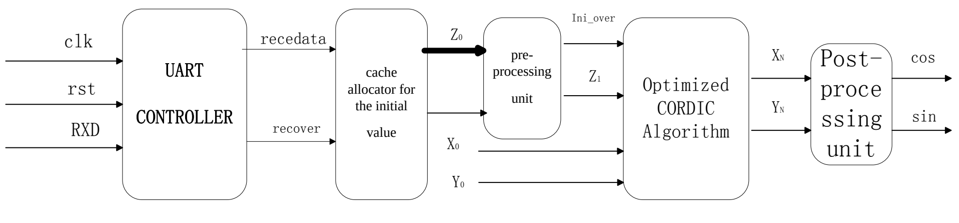

Figure 4. Overall Structure of the System

The UART controller is used to optimize the communication between the hardware of CORDIC and the serial device. The cache allocator is used to convert the two 8-bit datum to a 16-bit data for the initial value of the pre-processing unit. The pre-processing unit is used to convert initial angle into the first quadrant, trigger the iteration computation of the unit of optimized CORDIC. In the five modules, the unit of optimized CORDIC is the core, which determined the performance of system.

The flow chart of the optimized CORDIC algorithm is shown in Figure 5.

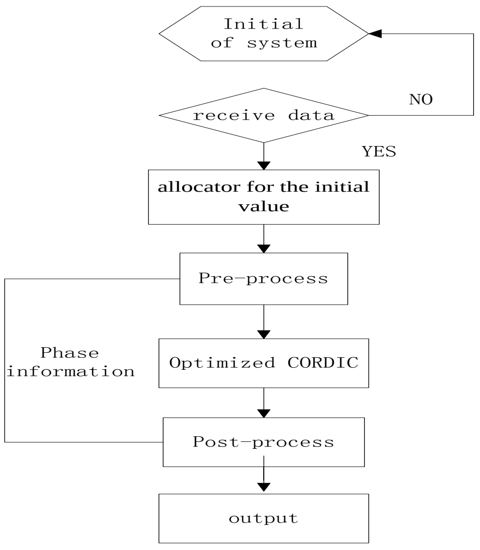

Figure 5. Overall Flow Chart System

5 .Simulation of CORDIC Algorithm based on FPGA

In this part, the results of simulation on FPGA both of traditional CORDIC algorithm and the optimized one

5.1 Simulation of Traditional CORDIC Algorithm

We take a 16-bit CORDIC algorithm as an example, chose the EPlC4F400C6 chip of the Cyclone series developed by Altera company, and select some angles randomly in the cycle, such as 15°, 45°, 99°, 110°, 200°. The angles between 0° and 360° are indicated by 16-bit binary and the result of the traditional CORDIC is shown in Fig. 6. Since the results are signed, the MSB (Most Significant Bit) is sign bit, and the remaining fifteen bits are the fractional part. The simulation results of sine and cosine function are shown in Table 6 and Table 7.

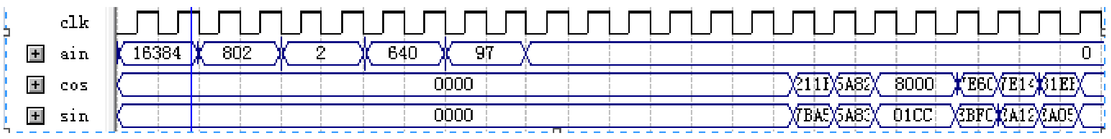

Figure 6. Result of the Simulation of Traditional CORDIC

Table 6. Sine Simulation Results of Traditional CORDIC

Angle	Exact value	Result of hex	Result of decimal	Inaccuracy
15°	0.25882	211F	0.25876	-6×10^{-5}-
45°	0.70711	5A82	0.70709	-2×10^{-5}

99.8°	0.98769	7E6C	0.98768	-1×10^{-5}
110°	0.93969	7E14	0.98492	4.527×10^{-2}
200°	-0.34202	81EE	-0.98492	6.429×10^{-1}

Table 7. Cosine Simulation Result of Traditional CORDIC

Angle	Exact value	Result of hex	Result of decimal	Inaccuracy
15°	0.96593	7BA5	0.96597	4×10^{-5} -
45°	0.70711	5A83	0.70712	1×10^{-5}
99.8°	-0.15643	EBFC	-015637	6×10^{-5}
110°	-0.34202	EA12	-0.17133	1.171×10^{-1}
200°	-0.93969	EA05	-0.17122	7685×10^{-1}

From Table 6 and Table 7, when the input angles are less than 99.8°, the inaccuracy of simulation negligible, otherwise the inaccuracy are large, which verify the contents we discussed above, and indicate that the conversion of input angle is necessary.

5.2 Simulation of the Optimized CORDIC Algorithm based on FPGA

We also take a 16-bit CORDIC algorithm as an example, chose the same chip and selected the same angle. And the angles are indicated with 16-bit unsigned binary, the results of simulation are shown as 16-bit complement, and the MSB is the sign bit, the remaining fifteen fits are decimal, the waveform diagram is shown in Fig. 7. The simulation results of sine and cosine function are shown in Table 8 and Table 9.

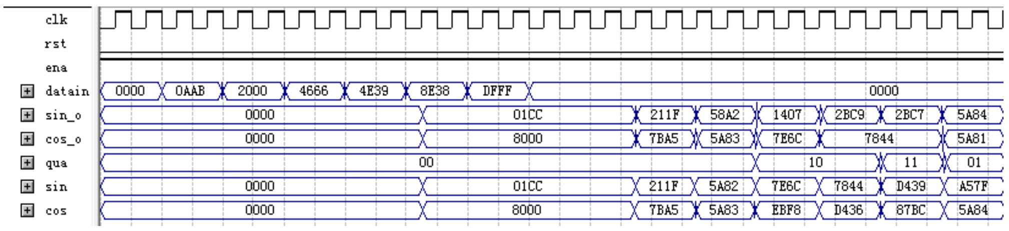

Figure 7. Result of the System

Table 8. Sine Simulation Results of the Optimized CORDIC

Angle	Exact value	Result without transform	Result with ransform	Inaccuracy
15°	0.25882	211F	211F	-6×10^{-5} -
45°	0.70711	5A83	5A82	-2×10^{-5}
99.8°	0.98769	1407	7E6C	-1×10^{-5}
110°	0.93969	2BC9	7844	-1×10^{-4}
200°	0.342022	279A	D439	0
315	-0.70711	5A88	A57F	4×10^{-5}

Table 9. Cosine Simulation Result of Optimized CORDIC

Angle	Exact value	Result without transform	Result with ransform	Inaccuracy
15°	0.96593	7BA5	7BA5	-7×10^{-5}
45°	0.70711	5A83	5A83	-1×10^{-5}

99.8°	0.15643	7E6C	EBF8	6×10^{-5}
110°	-0.34202	7844	D436	-8×10^{-5}
200°	-0.93969	79B6	87BC	1×10^{-4}
315	0.70711	5A7D	5A84	5×10^{-5}

From the results, the optimized CORDIC have a high accuracy as while as enhance operating frequency.

6. Conclusions

In this paper, we successfully complete the optimization of conventional CORDIC algorithm, and resolve the problem of restrictive relationship of speed, area, precision in the design, break the limitation of angle coverage, provide a optimization for various functions by CORDIC algorithm, and accomplish the simulation of the optimized CORDIC algorithm with 16-bit on FPGA. Comparing the simulation results of optimized CORDIC algorithm and traditional one, we can get the conclusion that the optimized CORDIC algorithm reduce resource consumption, and increased the maximum operating frequency, the accuracy of the CORDIC algorithm is 10-5 as same as the data of traditional one.

Acknowledgements

This work is supported by Science and Technology Research Funds of Education Depa rtment in Heilongjiang Province under Grant Nos. 12541174.

References

[1] C. S. Wu, A. Y. Wu and C. H. Lin, "high-performance/ low-latency vector rotationalCORDICarchitecture based on extended elementary angle set and trellis-based searching schemes." IEEE Transactions on Circuits System II：Analog Digital Signal Processing, (**2003**), vol. 50, no. 9, pp. 589-601.

[2] X. Hu, R. G. Harber and S. C. Bass." Expanding the range of convergence of t he CORDIC algorithm". IEEE Transaction on Computers, (**1991**), vol. 40, no. 1, pp. 13-21.

[3] K. Maharatna, S. Banerjee, E. Grass, M. Krstic and A. Troya, "Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture", IEEE Transaction on Circuits Systems for Video Technology, (**2005**), vol. 15, no. 11, pp. 1463-1474.

[4] F. J. Jaime, M. A. Sánchez and J. Hormigo, "Enhanced Scaling-Free CORDIC."IEEE Transactions On Circuits and Systems", Regular Papers, (**2010**), vol. 57, no. 7, pp. 1654-1662.

[5] H. Y. Hu, "The quantization effects of the CORDIC algorithm[J]", IEEETrans, on Signal Processing, vol. 40, (**1992**), pp. 834—844.

[6] T-B. Juang and M.-Y. Tsai, "Para—CORDIC: Parallel CORDIC Rotation Alg—orithm[J]", IEEE Transactions on Circuits and Systems, vol. 51, no. 8, (**2004**), pp. 1515—1524.

[7] K H. Bed and R E. Siferd, "VLSI implementations of low--power leading—one detector circuits", Proceedings of the IEEE SoutheastCon2006, (**2006**), pp. 279—284.

[8] J. M. P. Langlois and D. Al-Khalili, "Hardware optimized direct digital frequency synthesizer architecture with 60-dBc spectral purity," IEEE International Symposium on, vol. 5, May (**2002**), pp. 361-364.

[9] B. D. Yang, J. H. Choi, S. H. Han, L. S. Kim, and H. K. Yu, "An 800-MHz low-power direct digital frequency synthesizer with anon-chip D/A converter," IEEE J. Solid-State Circuits, vol. 39, no. 5, May (**2004**), pp. 761-774.

[10] M. A. Butt and S. Masud, "FPGA based bandwidth adjustable all digital direct frequency synthesizer," IEEE, Communications and Information Technology, 2009. ISCIT 2009. 9th International Symposium on, (**2009**), pp. 1399 – 1404.

[11] M. Genovese and E. Napoli, "Direct Digital Frequency Synthesizers implemented on high end FPGA devices," IEEE, Ph D. Research in Microelectronics and Electronics (PRIME), 2013 9th Conference on, 2013, pp. 137 – 140.

[12] Y.-J. Cao, Y. Wang and T.-Y. Sung, "A ROM-less direct digital frequency synthesizer based on a scaling-free CORDIC algorithm," IEEE Conference Publications, Strategic Technology (IFOST), 2011 6th International Forum on, (**2011**), pp. 1186 – 1189.

[13] Chimakurthy, L. S. J. Ghosh, M. Dai, F.F. Jaeger, R.C, "A novel DDS using nonlinear ROM addressing with improved compression ratio and quantization noise," IEEE Journals & Magazines, Ultrasonics, Ferroelectrics and Frequency Control, (**2006**), pp. 274 – 283.