ISSN: 2414-1895 DOI: 10.6919/ICJE.202208_8(8).0113

Optimization Design of CORDIC Algorithm based on Vivado HLS

Feng Xiao, Dongcai Xie, Xing Gan

The College of Nuclear Technology and Automation Engineering, Chengdu University of Technology, Chengdu 610059, China

Abstract

Aiming at the shortcomings of traditional CORDIC algorithm for calculating trigonometric functions and taking up more resources, this paper studies a hardware acceleration method for CORDIC algorithm to calculate angle sine and cosine using Vivado HLS (High-Level-Synthesis). By constraining the generation parameters of RTL, the purpose of reducing computing delay and saving on-chip resources is achieved. The test results show that the CORDIC algorithm optimized based on Vivado HLS can achieve lower clock delay and lower resource utilization than the traditional implementation with the same number of iterations.

Keywords

CORDIC Algorithm; Trigonometric Function Calculation; Hardware Acceleration.

1. Introduction

The Coordinate Rotation Digital Computer (CORDIC) was first proposed by J.Volder in 1959[1], and realized addition and shift instead of multiplication by preset rotation angle values. Compared with the direct multiplication calculation in the digital circuit, the shift addition operation is easier to implement, thereby reducing the circuit scale and the implementation cost. In 1971, J.S.Walther integrated the circular coordinate system, hyperbolic coordinate system and plane coordinate system into the same CORDIC iterative equation[2], so that the algorithm can calculate more functions. However, the traditional CORDIC algorithm is a serial algorithm, and the calculation accuracy is proportional to the number of iterations. The higher the output precision, the more iterations are required, which inevitably brings about the problem of slow calculation speed. This paper uses Vivado HLS to implement CORDIC hardware acceleration, which has good versatility, low computing delay, and low hardware resource consumption.

2. The Basic Principle of CORDIC Algorithm

The basic principle of calculating the sine and cosine of the CORDIC algorithm is shown in Figure 1.

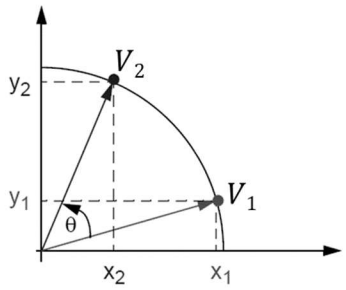

Figure 1. Coordinate rotation

ISSN: 2414-1895 DOI: 10.6919/ICJE.202208_8(8).0113

Assuming that the rotation angle of the vectors V1 to V2 is θ, then the conversion relationship between V2 and V1 is:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{1}$$

Equation (1) can be rewritten as:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -\tan\theta_n \\ \tan\theta_n & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{2}$$

Let.

$$\theta_n = \arctan(2^{-n}) \tag{3}$$

To achieve using shifts instead of computing $\tan\theta_n$.

$$\cos\theta_n = \sqrt{\frac{1}{1+2^{-2n}}} \tag{4}$$

Adding d_n represents the rotation direction of the vector, $d_n = +1$ represents counterclockwise rotation, $d_n = -1$ represents clockwise rotation, then formula (2) can be expressed as:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \sqrt{\frac{1}{1+2^{-2n}}} \begin{bmatrix} 1 & -d_n 2^{-n} \\ d_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{5}$$

in the formula, $\sqrt{\frac{1}{1+2^{-2n}}}$ is the correction factor, for operations with a fixed number of iterations, the product of the correction factors is also a fixed value. Therefore, the correction factor can be multiplied again after the iteration is over. So formula (5) can be expressed as:

$$\begin{cases} x_{n+1} = x_n - S_n 2^{-n} y_n \\ y_{n+1} = y_n + S_n 2^{-n} x_n \end{cases} \tag{6}$$

It can be seen from the above formula that the algorithm only has addition and shift operations, which is easy to implement in hardware circuits. After completing the iterative formula for the above coordinate calculation, an angle accumulator needs to be introduced to record the angle turned during the iterative process, which is expressed as:

$$z_{n+1} = z_n - d_n\theta_n \tag{7}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202208_8(8).0113

Among them, z_0 is the angle θ to be calculated. When z_n is positive, $d_n = 1$; when z_n is negative, $d_n = -1$. Finally, z_{n+1} is approximated to 0 within a certain accuracy range. The expression after the final iteration is as follows:

$$\begin{cases} x_n = K_n(x_0 \cos\theta - y_0 \sin\theta) \\ y_n = K_n(y_0 \cos\theta + x_0 \sin\theta) \\ z_n = 0 \end{cases} \tag{8}$$

Among them, K_n is the cumulative correction factor:

$$K_n = \prod_{i=0}^{n} \frac{1}{\cos\theta_i} = \prod_{i=0}^{n} \sqrt{1 + 2^{-2n}} \tag{9}$$

Let $x_0 = 1/K_n$, $y_0 = 0$, thus $\cos\theta = x_n$, $\sin\theta = y_n$.

3. Vivado HLS Implementation

3.1 Introduction to Vivado HLS

Vivado HLS is a high-level FPGA design tool launched by Xilinx. The tool can compile and synthesize the algorithm module described in the C standard language into an RTL hardware IP model for use in the Vivado development tool or System Generator. The developer only needs to focus on the implementation of the algorithm, and the Vivado HLS tool automatically considers the micro-architecture of the FPGA[3].

Vivado HLS optimizes the hardware implementation according to 3 parameters: the data initialization interval represents the clock interval between two consecutive new input data received by the module, a smaller II means greater data throughput; latency refers to the module The new data gives the number of delay clocks for all output data, and a smaller Latency indicates a faster computing speed; the area refers to the logic resources occupied by the module.

3.2 Top Module Design

The top-level design includes three sub-functions, which respectively implement angle preprocessing, calculation and post-processing. Since the calculable angle range of the traditional CORDIC algorithm is within the first quadrant, the preprocessing part needs to mirror the input angle value to the first quadrant; the calculation part implements the iterative algorithm of formula (6) through a for loop to calculate the sine and cosine values; The post-processing part needs to calculate the sine and cosine values of the angle after the mirror image of the pre-processing part, and then inversely calculate the sine and cosine values corresponding to the real input angle according to the corresponding relationship of the trigonometric function.

3.3 Comprehensive Realization

Vivado HLS provides a wealth of comprehensive optimization strategies, users can use different strategies to adjust and control internal logic and I/O behavior to take full advantage of hardware parallelism and obtain higher performance. According to the characteristics of the algorithm, the following optimization strategies are carried out.

The PIPELINE optimization strategy decomposes an operation that needs to be performed in multiple steps into several small operations that are executed in parallel. Each small operation takes less time and allows operations to occur simultaneously, so it can improve data throughput; usually, the loop is in Vivado HLS. The implementation is to use the same generated hardware resources in time-

ISSN: 2414-1895 DOI: 10.6919/ICJE.202208_8(8).0113

sharing. Although the resource utilization rate is high, it will also hinder the processing speed of the entire algorithm. The UNROLL optimization strategy can unroll the loop into multiple identical units to reduce data latency. Table 1 shows the comparison between the data delay and resource utilization report synthesized by the above optimization strategy and the implementation of the traditional CORDIC algorithm.

Table 1. Comparison of area consumption and latency between traditional way and Vivado HLS

Method to implementation	Resource		Latency	Interval
	LUT	FF		
Traditional	1086	1097	21	1
Vivado HLS	758	412	13	1

4. Conclusion

Using the Vivado HLS tool for the hardware acceleration design of the algorithm can avoid the mining of the underlying details in the traditional FPGA algorithm development, and effectively shorten the development cycle. This paper uses this tool to realize the optimal design of the CORDIC algorithm. Compared with the traditional implementation, the hardware circuit generated by Vivado HLS can run faster, have lower latency, and consume less resources.

References

[1] Volder J E. The CORDIC trigonometric computing technique [J]. IRE Transactions on Electronic Computers, 1959(8):330-334.

[2] Walther J. A unified algorithm for elementary functions[C]. Proceedings of the Spring Joint Computer Conference of the American Federation of Information Processing Societies (AFIPS). Atlantic City,NJ,USA:IEEE,1971:379-385.

[3] Xilinx, Inc.Vivado design suite user guide:high-level synthesis[Z].UG902, 2021.