一种降低 CORDIC 算法截断误差的方法

曲世隽，王翾

（中国传媒大学 信息与通信工程学院，北京 100024）

摘要：坐标旋转数字计算机（Coordinate Rotation Digital Computer,CORDIC）算法是一种使用坐标旋转执行一系列迭代计算的算法，仅通过加法器和移位器就可以实现复杂函数的运算。CORDIC 算法会在迭代计算中产生截断误差，这是影响 CORDIC 算法精度的重要原因。为了降低截断误差，提出了一种处理迭代计算中截断问题的方法。该方法通过少量增加硬件资源的消耗以大幅提高计算的精度。以 CORDIC 算法在圆坐标系矢量模式中的应用为例，给出了基于此方法的 CORDIC 算法的整体流程和硬件架构设计，完成在 ModelSim 平台上的仿真测试，将仿真结果与理论值进行比较分析，验证了方法的正确性和可行性。

关键词：CORDIC；截断误差；方法；ModelSim

中图分类号：TP302　　**文献标识码：A　文章编号：**

A method to reduce truncation error of CORDIC algorithm

QU Shijun, WANG Xuan

(School of Information and Communication Engineering, Communication University of China, Beijing 10024, China)

Abstract: CORDIC(Coordinate Rotation Digital Computer) algorithm is an algorithm that uses coordinate rotation to perform a series of iterative calculation. Complex function operation can be realized only by adder and shifter. CORDIC algorithm will produce truncation error in iterative calculation, which is an important reason affecting the accuracy of CORDIC algorithm. In order to reduce the truncation error, a method is proposed to deal with the truncation problem in iterative computation. This method greatly improves the calculation accuracy by increasing the consumption of hardware resources. Taking the application of CORDIC Algorithm in the vector mode of circular coordinate system as an example, the overall process and hardware architecture design of CORDIC algorithm based on this method are given. The simulation test on ModelSim platform is completed. The correctness and feasibility of the method are verified by comparing the simulation results with the theoretical values.

Key words:CORDIC;truncation error;method;ModelSim

作者简介：曲世隽（1997 - ），男（汉族），山东东营人，中国传媒大学硕士研究生，qushijun@cuc.edu.cn

1 引言

坐标旋转数字计算机（Coordinate Rotation Digital Computer,CORDIC）算法是一种用于高效计算基本函数的迭代算法。该算法通过一般的加减和移位操作实现了乘除法的相关计算，使得向量的的旋转和定向的计算不再依赖于三角函数、乘法、开方、反三角、指数等复杂函数。CORDIC 算法的硬件实现只需要移位器和加法器，简单的硬件就能有效的运算复杂函数。

在 1959 年，Volder 开发了一类计算三角函数、双曲函数的算法，其中包括指数运算和对数运算，这就是 CORDIC 算法的雏形。在提出这个算法之后，Volder 将其应用在导航领域之中。但是在当时这个算法并没有被广泛的讨论和应用。直到 1980 年，Haviland 和 Tuszynski[1]给出了一种全模式的 CORDIC 算法的处理芯片。在这之后，CORDIC 算法才引起人们的关注，被应用于各种领域。Hu Y H[2]将 CORDIC 算法应用于基于 VLSI 的数字信号处理领域。应用包括离散傅里叶变换的计算、矩阵运算和三角函数发生器。在 2009 年，Vachhani L,Sridharan K,Meher P K[3]通过 FPGA 实现高效 CORDIC 算法并将其应用在机器人探测上。当然，CORDIC 算法也可以应用于其他领域。例如，计算机制图中求点到线的距离、直角坐标与极坐标的相互变换及求多维向量的欧几里得范数等，可以预见，CORDIC 算法将在未来有更广泛的应用。

目前人们对于 CORDIC 算法的关注点主要集中在硬件资源消耗和计算精度两个问题上。想要获得高计算精度就必然要消耗更多的硬件资源，因此如何平衡两者关系就成了一个重要问题。在文献[4]中，提出了一种在硬件资源和计算精度之间寻求折中的方法。文中介绍了两种解决 CORDIC 算法精度问题的方法，第一种建立在定点数 CORDIC 算法的运算基础上，文中认为输入变量的不规范会使误差变大，因此需要对输入的变量进行统一、规范的处理，以此提高 CORDIC 算法的精度，第二种方法是针对于浮点数的 CORDIC 算法，通过使用混合体系结构降低复杂度，以此提高精度。在实际应用中，为了硬件结构的简单和运算的高效，我们更加偏向于使用定点数进行 CORDIC 算法。但是文中介绍的提高定点数精度的 CORDIC 算法在硬件结构上还是过于复杂，因此需要提出一种硬件结构简单，更容易实现的方法来降低误差。

为了提出一种更好的方法实现硬件资源和计算精度的平衡，就需要对 CORDIC 算法的误差来源进行详细的分析。CORDIC 算法的误差可以分为近似误差和截断误差，在任何坐标系的任何模式中，近似误差是由有限的迭代次数产生的，而截断误差是由有限的数据位位宽产生的[5]。Tze-Yun Sung,Yi-Hsun Sung[6]对所有坐标系下 CORDIC 算法的误差都进行了分析。包括圆坐标系，双曲坐标系，线性坐标系的旋转模式和矢量模式，用近似的方式给出了一个误差的最大值。但是文献主要是基于数学分析的角度，给出了近似误差和截断误差的分析结果。在实际 CORDIC 算法的应用中，需要有硬件结构的设计和反正切、正余弦函数的计算结果，这些在文章中没有提到。

本文提出了一种降低 CORDIC 算法中截断误差的方法，将 CORDIC 算法的流程分为预处理、迭代移位、后处理三个部分，对每个部分进行不同的处理，旨在使用较少的硬件资源同时保持较高的计算精度。文中给出了基于此种方法的硬件架构设计，并做了基于 FPGA 的仿真实验，验证了结果的可行性和正确性。

2 CORDIC 算法原理

CORDIC 算法的基本公式如下所示[5]：

$$x(i+1)=x(i)+m\delta(i)y(i)2^{-s(m,i)} \tag{1}$$

$$y(i+1)=y(i)-\delta(i)x(i)2^{-s(m,i)} \tag{2}$$

$$z(i+1)=z(i)-\delta(i)\alpha_m(i) \tag{3}$$

其中 $s(m,i)$ 是非递减的移位序列，满足

$$s(m,i)\le s(m,i+1)\le s(m,i)+1 \tag{4}$$

$$\alpha_m(i)=m^{-1/2}\tan^{-1}[m^{-1/2}2^{-s(m,i)}] \tag{5}$$

是第 i 次旋转的角度。$m=+1,-1,0$ 确定坐标系是圆，双曲或是线性。$\delta(i)$ 确定旋转角的方向，+1 和-1 分别代表不同的方向。由于 CORDIC 算法的实现方式很多，以圆坐标系为例，分旋转模式和矢量模式进行原理分析。

通常在旋转模式下，设定 $x(0)=1/k_1$，其中 $k_m=\prod_{i=0}^{n-1}\sqrt{1+m2^{-2i}}$，n 为迭代次数。$y(0)=0$。$z(0)$ 为输入待计算的角度 θ。旋转方向 $\delta(i)$ 由 $z(i)$ 的正负决定，若 $z(i)$ 为正，则 $\delta(i)$ 为+1，若 $z(i)$ 为负，则 $\delta(i)$ 为-1。根据公式(1)(2)(3)(5)可以得到角度 θ 的正弦值 $y(n)$ 和余弦值 $x(n)$。

通常在矢量模式下，设定输入的值 x 和 y，通过迭代和旋转，使 $y(i)$ 逐渐逼近于 0。旋转方向 $\delta(i)$ 由 $y(i)$ 的正负决定，若 $y(i)$ 为正，则 $\delta(i)$ 为-1，若 $y(i)$ 为负，则 $\delta(i)$ 为+1。同样根据公式(1)(2)(3)(5)，可以得到反正切函数 $z(n)=\arctan(y/x)$，得到向量 (x,y) 的长度 $d=x(n)*(1/k_1)$。

3 CORDIC 算法的误差

在研究 CORDIC 算法的时候，我们需要在硬件资源的使用和计算精度上做出平衡，这就需要对 CORDIC 算法的误差进行分析。CORDIC 算法的误差由迭代次数 n，旋转角度小数位位数 b，x 和 y 的小数位位数 b_v 确定。在本文中 b 和 b_v 采用相同的大小。CORDIC 算法的误差可以分为两部分：近似误差和截断误差。E_A 表示近似误差最大值，由有限的迭代次数造成，E_R 表示截断误差最大值，由有限的数据位位宽造成。

根据文献[6]，圆坐标系下旋转模式的近似误差为

$$E_A=\tan^{-1}2^{-(n-1)}+n\cdot 2^{-b} \tag{6}$$

截断误差为

$$E_R=\sqrt{2}\cdot 2^{-b_v}[1+\sum_{j=1}^{n-1}(\prod_{i=j}^{n-1}\sqrt{1+2^{-2i}})] \tag{7}$$

矢量模式下的近似误差为

$$E_A=\sin^{-1}(\tan^{-1}2^{-(n-1)}+n\cdot 2^{-b}) \tag{8}$$

截断误差为

$$E_R=n\cdot 2^{-b} \tag{9}$$

基于以上的理论推导，可以确定迭代次数和小数位位数对于 CORDIC 算法的误差的影响，如图 1 和图 2 所示。

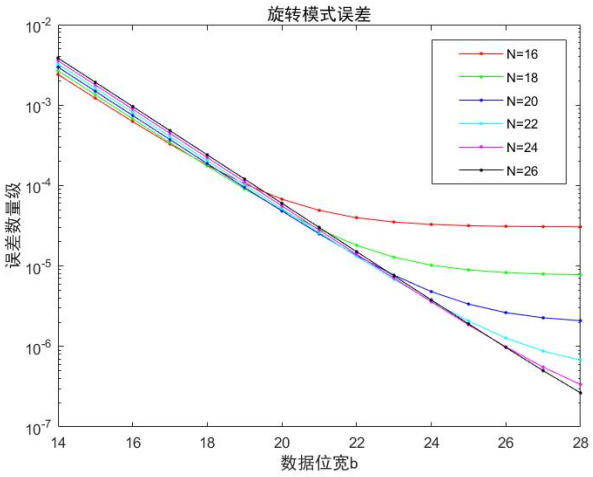

图 1 圆坐标系旋转模式误差

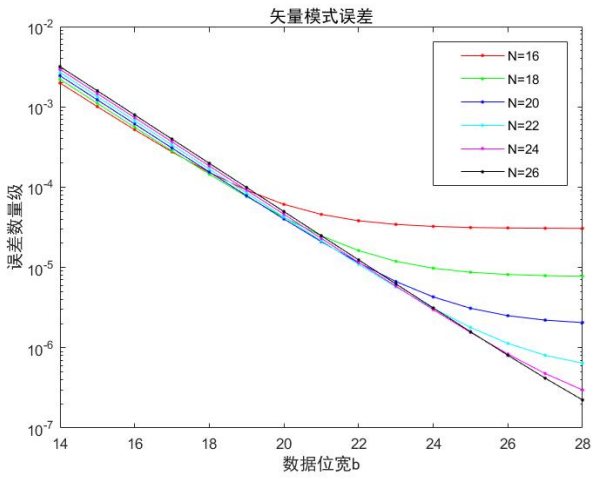

图 2 圆坐标系矢量模式误差

4 降低截断误差的方法

4.1 方法简述

从前文的分析中可以得知，增加数据位位宽和迭代次数可以显著的提高 CORDIC 算法的精度，但是同时也会消耗更多的硬件资源。降低近似误差，只需要增加迭代的次数，但是降低截断误差却不能只是简单的增加数据位位宽。过于长的数据位位宽不仅会消耗更多的硬件资源，同时也会对后续的数据处理造成影响。因此，需要有一个合适的方法来降低截断误差。

通过此种方式实现 CORDIC 算法，完整的步骤可以分为预处理、迭代移位、后处理三个部分。三个步骤如图 3 所示。

图 3 实现改进 CORDIC 算法完整步骤

为了有更高的精度，我们选择在预处理部分进行数据的规范化处理和扩充数据位位宽。在迭代移位的过程中如果改变数据位位宽，很可能会产生溢出问题，因此选择在后处理部分对迭代移位后的数据进行截断。这样就可以保证在迭代移位部分提高了计算精度，同时输出位宽又可以根据实际的需要进行截取。

4.2 预处理

预处理部分可以分为对输入数据进行规范化处理和对输入数据的位宽进行扩充两个模块。假设输入数据 x_in 和 y_in 为 16bit 定点数，最高位为符号位。对输入数据进行规范化处理要计算除符号位之外的先导零的个数。比较 x_in 和 y_in 的先导零个数，j 为两者先导零个数的较小值，根据公式

$$j=\left\lfloor \log_2\left[\min\left\{\frac{2^{m-1}\varepsilon}{|x\_in|},\frac{2^{m-1}\varepsilon}{|y\_in|}\right\}\right]\right\rfloor \tag{10}$$

可以得到 j 的值。其中 m 为输入数据位宽，$\varepsilon=2^{-b}$ 为精度。在硬件设计中，可以通过一个 15-4 优先编码器和一个逐位比较的或门来得到 j 的值。x_in 和 y_in 并行输入逐位比较的或门，可以得到两者之间较高位的 1 所在的位置。再通过 15-4 优先编码器得到较高位的 1 之前先导零的个数。在得到 j 之后，统一将 x_in 和 y_in 左移 j 位。为了防止迭代过程中发生溢出，还要在符号位后增加两个 0 位。根据计算，可以得到 k_1 的值在 1.414 和 1.646 之间，因此扩充两个 0 位完全可以避免溢出。15-4 优先编码器的真值表如表 1 所示。

表 1 15-4 优先编码器真值表

输入值	输出值
1 XXXXXXXXXXXXXX	0000
0 1 XXXXXXXXXXXXX	0001
.....	.....
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1111

对输入数据的位宽进行扩充就是在输入数据末尾补零。我们将输入数据扩充为 32bit。也就是在进行完数据的规范化处理之后需要在末尾补 14 位的 0。预处理硬件设计如下图 4 所示。

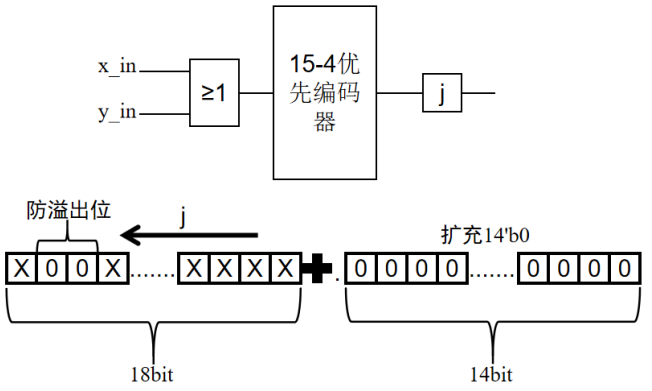

图 4 预处理硬件设计

4.3 迭代移位

CORDIC 算法的单层迭代移位结构如图 5 所示。它需要 1 个多路复用器，3 个加/减法器，2 个移位器。旋转方向由 *selx* , *sely* , *selz* 定义，多路复用器根据旋转模式或矢量模式来进行选择。当模式为旋转模式时，旋转方向由 $\mathrm{sign}(z_i)$ 确定。当模式为矢量模式时，旋转方向由 $\mathrm{sign}(y_i)$ 确定。要实现并行流水线的 CORDIC 算法，只需要对单层的结构进行叠加就可以实现。叠加的次数取决于迭代次数 N。

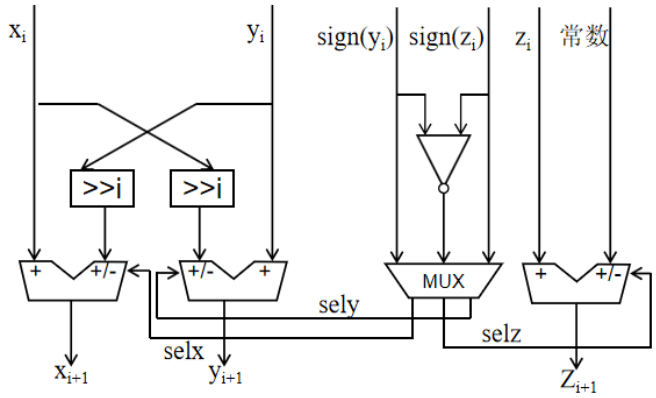

图 5 单层迭代结构

4.4 后处理

后处理的部分实际上就是对迭代移位后输出的数据进行位宽的缩减，本质上也是对前导零的去除。缩减的原则是，首先从符号位后最高位开始判断，如果最高位为 0，则向左移 1 位，继续判断下一位，若还为 0，则继续向左移 1 位，判断下一位，直到遇到为 1 的位数，则停止判断。对 0 的位数舍去完之后，整体保留 16bit。设后处理部分总共左移的位数为 *num* 。后处理部分流程图如图 6 所示。

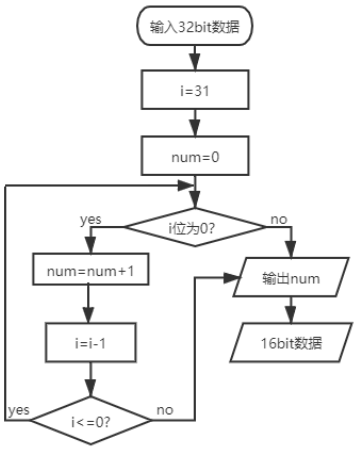

图 6 后处理流程图

通过流程图可知，实现后处理部分需要通过循环结构来进行判断，为了节省硬件资源，可以使用计数器来实现循环结构。因此，后处理部分需要 1 个累加

器和 1 个计数器，1 个寄存器。将需要处理的数据放入寄存器，在每个计数器+1 时进行逐位判断。后处理结构如图 7 所示。

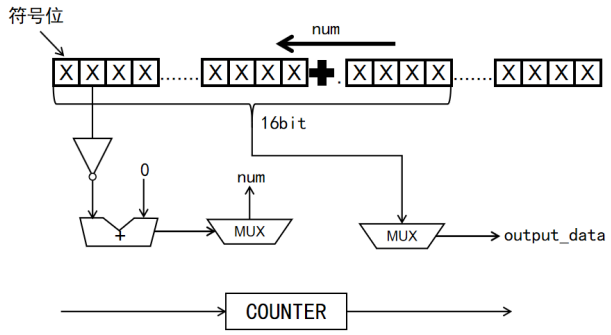

图 7 后处理结构

最后可以总观整个流程中对数据的左移，可以得到数据在预处理部分左移的位数为$14+j$,在后处理部分左移的位数为$num-16$。所以整个流程对数据的放大为$2^{j+num-2}$倍。也就是说得到的数据精度为$\varepsilon=2^{j+num-2}$,在改进后的 CORDIC 算法中，可以在预处理和后处理部分去除无意义的前导零，以此提升精度。

5 仿真与测试

在本设计中采用 Spartan 3E 开发板，用 Verilog 对 CORDIC 算法在圆坐标系下的矢量模式进行了描述，采用流水线结构。对于 100000 组输入范围在(0,10000)的独立均匀分布的 16bit 数据进行基于 FPGA 的圆坐标系矢量模式下的 CORDIC 算法计算。选择迭代次数为 16 次，在预处理时将其扩充为 32bit，在后处理时保留 16bit。为了验证其正确性，对其实现过程用 ModelSim10.5 进行仿真，仿真结果如图 8 所示。

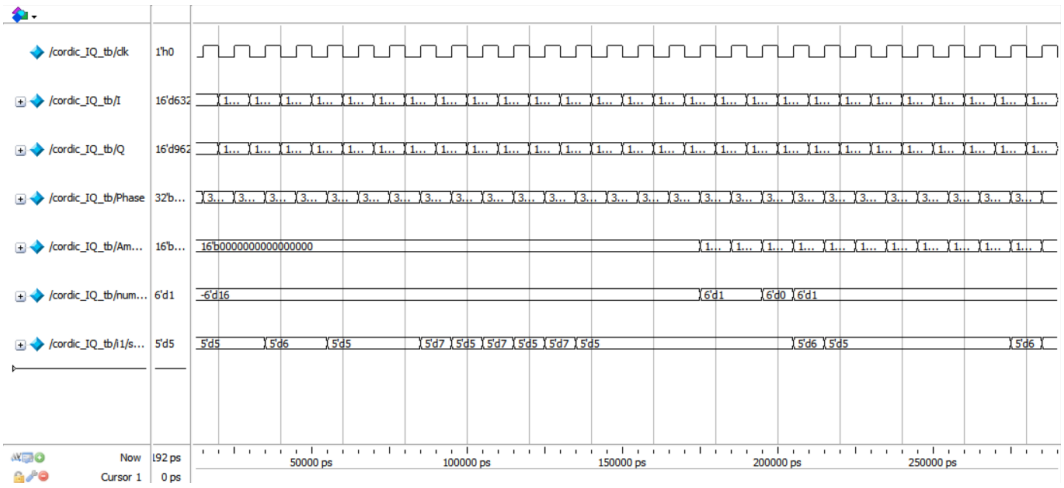

图 8 ModelSim 仿真结果

从仿真结果中可以看出输入和输出结果之间存在延时，这跟预处理和迭代移位过程有关。

提取仿真过程中激励文件随机生成的 100000 组输入值，对比理想输出相位值、幅度值和实际输出相位值、幅度值之间的绝对误差。其绝对误差的分布如图 9 所示。

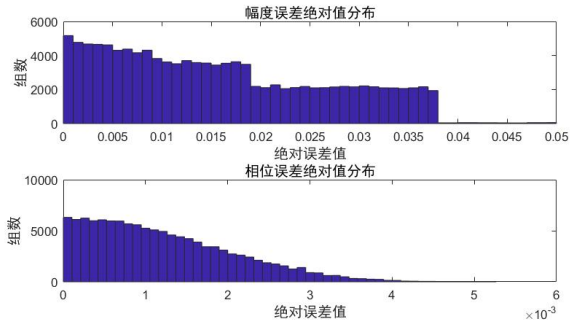

图 9 使用截断方法的 CORDIC 算法的绝对误差分布

计算得到幅度值的平均绝对误差为 $1.67*10^{-2}$，方差为 $1.3956*10^{-4}$。相位值的平均绝对误差为 $1.1*10^{-3}$，方差为 $2.0246*10^{-6}$。对比使用相同数据位位宽的传统 CORDIC 算法，在精度上有了一定的提升。

6 结束语

该文简要介绍了 CORDIC 算法的原理，从硬件实际应用角度出发，提出了一个降低 CORDIC 算法截断误差的方法。实现了对复杂函数的高精度运算，并给出了圆坐标系矢量模式下的仿真结果。在实验过程中，可以通过改变迭代的次数或是数据位位宽来满足不同的精度要求。从仿真结果来看，提出的方法可以很好的在精度要求和硬件资源消耗上达到平衡。这个方法在本文中只给出了在圆坐标系矢量模式的应用分析，在以后的研究中，可以将这个方法推广到圆坐标系、线性坐标系、双曲坐标系的不同模式中。为 CORDIC 算法计算开方、反三角等复杂函数问题提供了一个新的处理方法，具有一定的借鉴意义。

参考文献

[1] Haviland, G L, Tuszynski A A, et al. A CORDIC arith metic processor chip[J]. IEEE Journal of Solid-State Circ uits,1980,15(1):4-15.

[2] Hu Y H. Cordic-based VLSI architectures for digital si gnal processing[J]. IEEE Signal Processing Magazine,19 92, 9(3):16-35.

[3] Vachhani L, Sridharan K, Meher P K. Efficient FPGA realization of CORDIC with application to robotic ex-plo ration[J]. IEEE Transactions on Industrial Electronics,200 9,56(12):4915-4929.

[4] Kota K, Cavallaro J R. Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special-purpose processors[J]. IEEE Transactions on Computers,1993, 42(7):769-779.

[5] Hu Y H. The quantization effects of the CORDIC algorithm[C]. International Conference on Acoustics, Speech, and Signal Processing, 1988.

[6] Sung T Y , Sung Y H . The Quantization Effects of CORDIC Arithmetic for Digital Signal Processing Applications[C]. The 21st Workshop on Combinatorial Mathematics and Computation Theory, 2004:16-25.

[7] Fang L, Li B, Xie Y, et al. A unified reconfigurable CORDIC processor for floating-point arithmetic[J]. International Journal of Electronics, 2018,107(9): 1436-1450.

[8] Aggarwal S, Meher P K, Khare K. Concept, design, and implementation of reconfigurable CORDIC[J]. IEEE Transactions on Very Large Scale Integration Systems, 2016, 24(4):1588-1592.

[9] Pongyupinpanich S, Glesner M. Pipelined floating-point architecture for a phase and magnitude detector based on CORDIC[C]. 21st International Conference on Field Programmable Logic and Applications, Chania, Greece, 2011.

[10] Yan W, Chen H, Xie Y Z, et al. A new chirp scaling algorithm using region-constant phase multipliers[J]. Transactions of Beijing Institute of Technology, 2014,34(3): 304-309.

[11] 景标, 刑维巍, 张燕琴. 基于 CORDIC 算法的正交信号源实现[J]. 现代电子技术, 2016,39(7):57-59.

[12] 黄如, 徐磊. 一种基于 FPGA 的 CORDIC 算法的实现[J]. 电子测量技术, 2018,41(5):47-50.

[13] 王强，应浩. 反正切函数的 CORDIC 算法及其 FPGA 实现[J]. 兵工自动化, 2020,39(6):45-48.

[14] 李翔. CORDIC 算法误差研究[J]. 科技创新导报，2017,(14):34-35.

地址：中国传媒大学主楼 109

姓名：曲世隽

邮箱：qushijun@cuc.edu.cn

电话：18562069596