文章编号:1003－6199(2024)01－0089－06 DOI:10.16339/j.cnki.jsjsyzdh.202401015

基于一种改进CORDIC算法的MATH处理器设计与优化

陈家敏,胡锦†

(湖南大学 物理与微电子科学学院,湖南 长沙 410082)

摘 要:在精密的电机数控领域中,对三角函数运算的硬件架构性能指标日渐严格。针对传统CORDIR算法求解三角运算时存在迭代次数多、迭代周期长、输入角度范围小等局限性,提出了对其进行角度预处理、镜像迭代、角度补偿、区间换算以及合并迭代结构等优化,并最终完成高精度计算三角函数的MATH处理器设计。在硬件实现上,本处理器在输入角度及坐标范围得到明显优化,计算速率显著倍增,且精度完全满足设计标准,适配于高精度电机驱动等应用领域。

关键词:CORDIC算法;角度预处理;镜像迭代;合并迭代结构;电机驱动

中图分类号:TN46 **文献标识码**:A

Design and Optimization of a MATH Processor Based on an Improved CORDIC Algorithm

CHEN Jiamin, HU Jin†

(School of Physics & Electronics, Hunan University, Changsha, Hunan 410082, China)

Abstract: In the field of precision motor numerical control, the performance indicators of the hardware architecture for trigonometric calculations are becoming increasingly strict. In view of the limitations of the traditional CORDIC algorithm for solving trigonometric operations, such as multiple iterations, long iterative cycles, and a small range of input angles, this paper proposes the optimization of angle preprocessing, mirror iteration, angle compensation, interval conversion, and merging iteration structures, thereby completing the design of a MATH processor for high-precision calculation of trigonometric functions. In terms of hardware implementation, this processor has significantly optimized the range of input angles and coordinates, markedly increased calculation rate, fully satisfies the design standard in terms of precision, and is suitable for applications in the field of high-precision motor drives.

Key words: CORDIC algorithm; angle preprocessing; mirror iteration; merged iteration structure; motor drive

在精密电机数控系统中存在着大量的三角函数运算,三角函数的求解通常会利用级数展开,其本质上是用多项式函数来近似三角函数,求解过程中不可避免地涉及大量的浮点数运算,这对缺乏硬件乘法器资源的设备来说是一项费时费力的任务。为了解决这个问题。J. E. Volder 在1957年发表的一篇论文中首次提到CORDIC算法,CORDIC算法是一种仅仅通过一系列简单的移位和加减操作就实现乘除法相关计算的迭代方法。

然而随着电机数控系统对计算精度要求越来越高,对CORDIR算法电路的性能指标也在不断提高,传统的CORDIC算法计算精度与迭代次数

收稿日期:2023－08－29
基金项目:湖南省重点研发计划资助项目(2021GK2004)
作者简介:陈家敏(1990—),男,湖南长沙人,硕士研究生,研究方向:数字集成电路设计。
† **通信联系人**,E-mail: hujin@hnu.edu.cn

成正相关，通常情况下，增加迭代次数可以提高算法的计算精度。每次迭代都会逐步逼近目标值，通过多次迭代，可以达到更高的精度要求，但是多次迭代会带来运算周期长等不足[1]，此外 CORDIC 算法还存在输出角度单一等缺陷，针对这些不足，国内外很多学者进行了研究，文献[2]通过自适应角度旋转的方法，减少了迭代周期，增加了算法灵活性，但是每次计算需额外判断，实现成本增加。文献[3]通过采用弧度制与角度制两种输出方法弥补输出角度单一的局限，但是仅支持向量模式改进有限。文献[4]选择多级合并结构和查找表结合，虽然性能有所提高，但是硬件资源消耗更大。本文对传统的 CORDIC 算法进行改进，提出角度预处理、镜像迭代、合并迭代结构、角度补偿与区间换算等措施，设计了一款基于 CORDIC 算法的 MATH 处理器，在有效地提高运算精度的同时减少迭代周期，能够满足电机控制系统对高精度三角函数运算的需求。

1 CORDIC 算法原理

CORDIC 算法通过连续的旋转操作逐步逼近目标角度，并根据每次迭代旋转的预设角度来更新计算结果。

1.1 迭代方程

首先，引入一个坐标位于直角坐标系的圆上，使得初始坐标为 (x_1,y_1)，经过一次角度的逆时针旋转，得到坐标 (x_2,y_2)，见图 1。

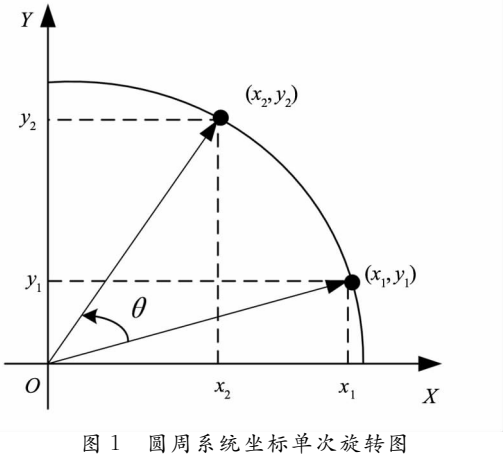

图 1 圆周系统坐标单次旋转图

由圆的坐标旋转公式：

$$\begin{cases} x_2 = x_1\cos\theta - y_1\sin\theta = \cos\theta(x_1 - y_1\tan\theta) \\ y_2 = x_1\sin\theta + y_1\cos\theta = \cos\theta(x_1\tan\theta + y_1) \end{cases} \tag{1}$$

CORDIR 的核心就是通过多次旋转预设角度来逼近目标角度，见图 2。

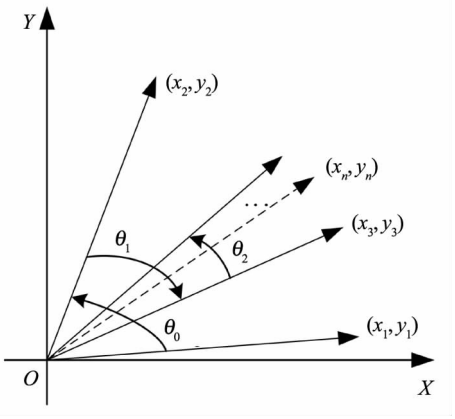

图 2 圆周系统坐标多次旋转图

图 2 中每一次旋转的方向用旋转方向位 d_i 来表示，坐标旋转是顺时针方向则 $d_i=-1$，坐标旋转是逆时针方向则 $d_i=1$，所以经过多次旋转的累积角度可以用公式(2)表示：

$$\theta = \sum_{i=0}^{n-1} d_i\theta_i \tag{2}$$

同时对于第 N 次旋转有：

$$\begin{cases} X_{N+1} = \prod_{i=0}^{N-1}\cos\theta_i\left(X_N - Y_N\tan\left(\sum_{i=0}^{n-1} d_i\theta_i\right)\right) \\ Y_{N+1} = \prod_{i=0}^{N-1}\cos\theta_i\left(Y_N + X_N\tan\left(\sum_{i=0}^{n-1} d_i\theta_i\right)\right) \end{cases} \tag{3}$$

公式(3)左右两边同时除以 $\prod_{i=0}^{N-1}\cos\theta_i$，得到伪旋转方程：

$$\begin{cases} X_{N+1} = X_N - Y_N\tan\left(\sum_{i=0}^{n-1} d_i\theta_i\right) \\ Y_{N+1} = Y_N + X_N\tan\left(\sum_{i=0}^{n-1} d_i\theta_i\right) \end{cases} \tag{4}$$

伪旋转方程目前还是存在乘法运算，容易联想到在二进制中，常常采用移位操作代替乘法运算，所以找到 $\tan\theta$ 与 2^{-i} 的对应关系，使得 $\tan\theta=2^{-i}$。

所以预设角度为固定的值，可以确定目标旋转角度的最大值和最小值：

$$\begin{cases} \theta_{\max} = \sum_{i=0}^{\infty} tan^{-1}(2^{-i}) \approx 99.883 \\ \theta_{\min} = \sum_{i=0}^{\infty} -\tan^{-1}(2^{-i}) \approx -99.883 \end{cases} \tag{5}$$

所以只要目标角度在范围 $(99.883°, -99883°)$ 之内，就一定能通过旋转多次逼近至目标角度，如表 1 所示。

表 1　预设旋转角度表

i	2^{-i}	$\tan^{-1}(2^{-i})$
0	1	45
1	2^{-1}	26.565
2	2^{-2}	14.036
3	2^{-3}	7.125
4	2^{-4}	3.576
5	2^{-5}	1.790
6	2^{-6}	0.895
7	2^{-7}	0.448

由公式(4)可知伪旋转方程每次旋转的角度是正确的，但是改变了坐标向量的模，为了保持旋转前后向量的模一致，新引入一个矫正因子：

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

校正因子只和迭代次数有关，当迭代次数足够多时，其收敛到 0.607，所以确定了迭代次数，矫正因子就可用常数来代替。该矫正因子在圆周系统的旋转模式下计算正弦函数 $\sin\theta$ 和余弦函数 $\cos\theta$ 时使用[5]。

引入角度累加方程，用于表示多次旋转后的累加到当前角度。得到最终的统一迭代方程：

$$\begin{cases} X_{i+1} = X_i - Y_i d_i 2^{-i} \\ Y_{i+1} = Y_i + X_i d_i 2^{-i} \\ Z_{i+1} = Z_i - \mathrm{d}_i\theta \end{cases} \tag{7}$$

1.2　圆周系统下的旋转模式

基于上述推导得到的迭代方程，输入一个初始角度 θ 可以计算正弦函数和余弦函数的函数值。原理是若多次旋转的向量坐标落在单位圆上，由于单位圆坐标满足 $X^2+Y^2=1$，同时 $\sin^2\theta+\cos^2\theta=1$，此时向量横坐标即为 $\cos\theta$，纵坐标为 $\sin\theta$。

所以首先设置初始坐标为 $(\frac{1}{K},0)$，输入角度为 $Z_0=\theta$，此时旋转方向 d_i 取决于累加角度 Z_{i+1}。

$$d_i = \begin{cases} 1, & Z_i \geqslant 0 \\ -1, & Z_i < 0 \end{cases} \tag{8}$$

经过多次预设角度迭代后，使得目标角度 $Z_{i+1}=0$，此时向量模长为 1 位于单位圆上，得到正弦函数 $\sin\theta=Y_{i+1}$ 和余弦函数 $\cos\theta=X_{i+1}$ 的函数值。

1.3　圆周系统下的向量模式

向量模式用于计算反正切函数 $\arctan\frac{y_0}{x_0}$ 的函数值。输入向量 (x_0,y_0) 经过多次迭代之后，最终使得向量与横轴 X 重合，也即 $Y_{i+1}=0$，此时旋转方向由 Y_i 决定。

$$d_i = \begin{cases} 1, & Y_i < 0 \\ -1, & Y_i \geq 0 \end{cases} \tag{9}$$

最终得到结果为 $Z_{i+1}=Z_0+\arctan\frac{y_0}{x_0}$，只要使初始角度 $Z_0=0$，就可以得到反正切函数 $\arctan\frac{y_0}{x_0}$ 的函数值

2　MATH 处理器的设计与优化

2.1　CORDIC 算法局限性优化

如果采用上文描述的 CORDIC 算法设计出来的处理器有非常明显的局限性，比如所提到目标角度的范围必须在 (99.883°，－99883°)，这意味着第二、三象限的绝大多数角度或者向量是不能通过 CIRDIC 算法来计算的。本文针对其局限性做出相应的改进[6]。

圆旋转模式：支持十六位无符号二进制数满范围 $[0,2^{16}-1]$ 内的初始角度 Z_0，代表 0 ～ 360°。提出使用角度预处理的方法首先对初始角度进行象限判断，并将非第一象限的初始角度区间映射在第一象限：

assign$Z_0op[15:0]=\{2'b00,Z_0[13:0]\}$；

通过多次迭代算出 sinZ_0op 和 cosZ_0op，然后对非第一象限的初始角度分别作 90°、180°、270° 补偿，由诱导公式可得表 2：

表 2　角度预处理与补偿

初始角度 Z_0	角度预处理 Z_0op	sinZ_0	cosZ_0
$[0,\frac{\pi}{2})$	Z_0	sinZ_0op	cosZ_0op
$[\frac{\pi}{2},\pi)$	$Z_0-\frac{\pi}{2}$	cosZ_0op	－sinZ_0op
$[\pi,\frac{3\pi}{2})$	$Z_0-\pi$	－sinZ_0op	－cosZ_0op
$[\frac{3\pi}{2},2\pi)$	$Z_0-\frac{3\pi}{2}$	cosZ_0op	sinZ_0op

最终输出的 sinZ_0 和 cosZ_0，按照 $Z_0[15:14]$ 选择对应的角度补偿输出，通过采用角度预处理以及角度补偿的方式，可以保证输入初始角度可以覆盖到所有象限角度，解决了 CORDIC 算法的输入旋转角度局限性。

圆向量模式：支持输入满范围 $[-(2^{15}-1),2^{15}$

$-1]$ 的初始向量坐标(x_0, y_0)来计算 $\arctan\frac{y_0}{x_0}$，输出结果为 $[0, 2^{16}-1]$，代表 $0 \sim 360°$。受 θ_{max} 和 θ_{min} 的限制，旋转向量的坐标限制在第一、四象限。在这种模式下，上述提到旋转方向取决于 Y_i，目标是使得 $Y_{i+1}=0$ 也即第一、四象限的向量多次向着直角坐标系横轴的正半轴迭代。为了解决这个问题，提出镜像迭代的方式，首先对输入向量坐标进行象限判断，若向量坐标位于二、三象限，则采用镜像迭代，见图 3。

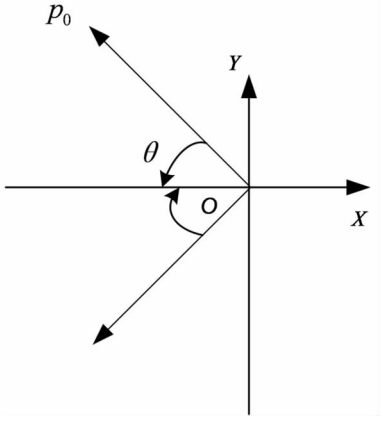

图 3 第二、三象限镜像迭代旋转图

如图 3 所示，采用镜像迭代的方式就可以把初始向量坐标覆盖到所有的象限，此时旋转方向不再取决于 Y_i 的值：

$$d_i=\begin{cases}1, & x_0[15]!=y_0[15]\\ -1, & x_0[15]==y_0[15]\end{cases} \tag{10}$$

由图中可知，旋转角度 θ 并非最后的结果，还需要进行区间换算才能得到准确的结果，见表 3。

表 3 区间换算

象限	$x_0[15]$	$y_0[15]$	$\arctan\frac{y_0}{x_0}$
一	0	0	θ
二	1	0	$\pi-\theta$
三	1	1	$\pi+\theta$
四	0	1	$2\pi-\theta$

任意向量坐标最大累加迭代角度不会超过 $\frac{\pi}{2}$，在保证了 CORDIC 算法的有效性同时，又有效地解决了初始向量坐标的局限性。

2.2 CORDIC 架构

CORDIC 算法是一种迭代式算法，需要进行多次循环才能得出最终结果，通常来说 CORDIC 算法是串行迭代架构，若搭载在 AHB 总线上，迭代 N 次需要消耗 N 个 HCLK 周期，若要求更高的精度，则需要继续迭代，累计的运算周期会导致运算时间过长，CORDIC 单次迭代结构如图 4 所示：

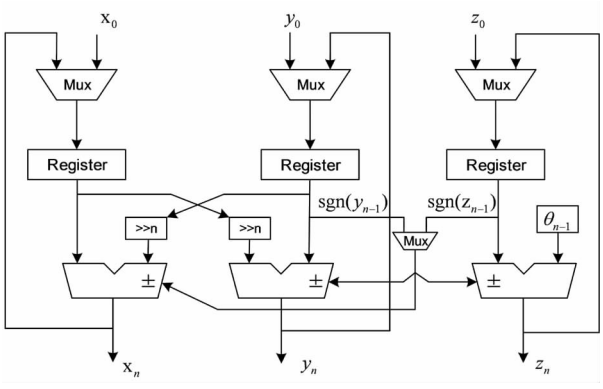

图 4 一次迭代结构图

串行迭代架构的利用率不高，主要是因为在高精度求解过程中迭代次数较多，迭代周期较长[9]。所以本文设计采用合并迭代架构，实现了在每个周期 CORDIC 的两次迭代，减少了迭代周期，提高处理器运算速度。如果需要进一步提高运算速度，只需要增加并行级数即可，但同时会带来资源占用及功耗增加等问题，所以具体采用哪种架构需要从速度、面积及功耗等方面综合考虑[10]，图 5 显示了 CORDIC 引擎合并迭代结构。

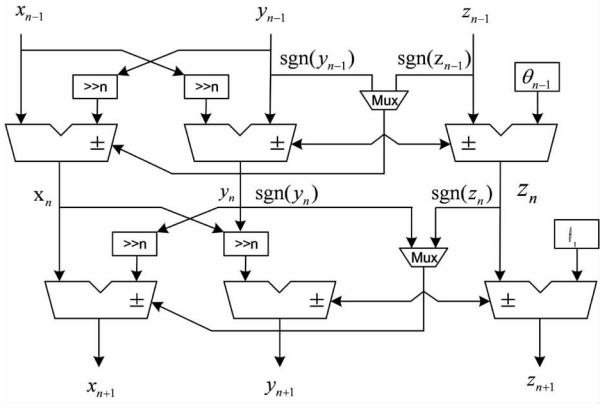

图 5 合并迭代架构图

3 MATH 处理器的实现与仿真

MATH 处理器采用 Verilog HDL 进行设计，电路结构包括几个部分：输入接口用于接收输入，如旋转角度和向量坐标；初始化模块负责对输入旋转角度、向量坐标进行角度预处理或者坐标象限判断；旋转迭代模块是核心部分，执行 CORDIC 算法的旋转操作包括迭代次数、方向控制、存储计算的预设角度、迭代计算逻辑等；数据转换模块对计算

结果做角度补偿或者区间换算，得到最终正确的结果并输入，其电路结构如图 6 所示。

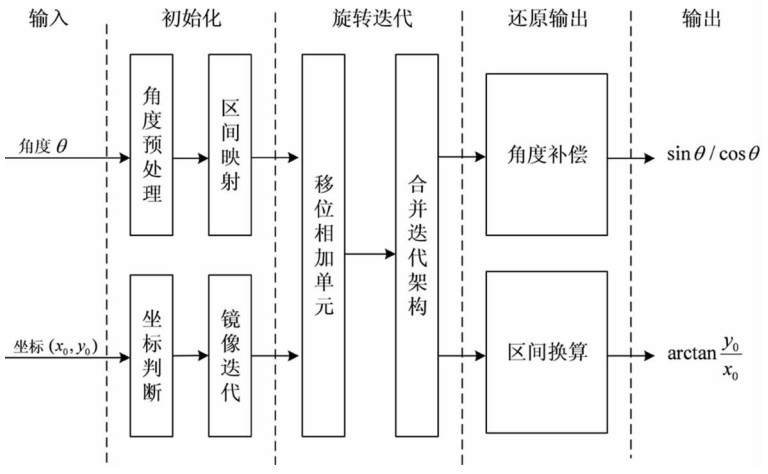

图 6　硬件结构框图

该处理器支持 AHB 从机接口，可搭载在 AHB 总线下，通过 AHB 总线时序启动该处理器并且配置相应的工作模式。若处理器采用传统的 CORDIC 算法的串行迭代架构，利用 VCS＋VERDI 进行联合仿真，处理器采用合并迭代结构，花费 N 个时钟周期可迭代 $2N$ 次，也即 12 个时钟周期便可完成 24 次迭代运算，迭代次数越多，则精度越高，运算效率提高了 100%，并行迭代结构仿真图见图 7。

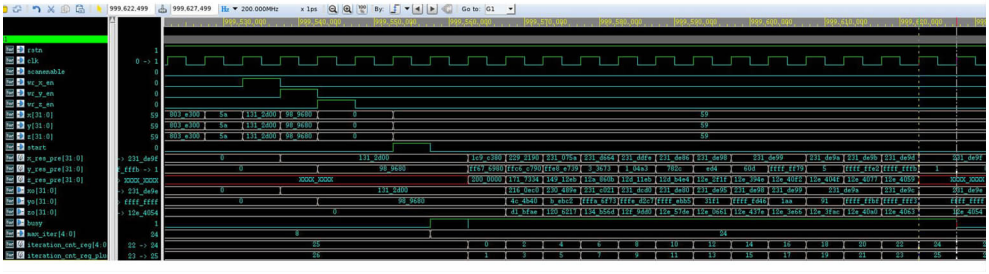

图 7　并行迭代结构仿真图

旋转模式下，支持十六位无符号二进制数满范围 $[0, 2^{16}-1]$ 内的初始角度 Z_0，代表 0 ～ 360°。输出结果范围为 $[-(2^{15}-1), 2^{15}-1]$ 的 sin、cos 值表示 $[-1, 1]$，实际计算结果和理论值的比较如表 4 所示：

表 4　旋转模式下计算结果与理论值的比较

输入角度 θ	$\sin\theta$ 输出值	$\cos\theta$ 输出值	$\sin\theta$ 理论值	$\cos\theta$ 理论值
16‘d8192	16‘d23172	16‘d23165	16‘d23170	16‘d23170
16‘d24576	16’d23165	16’d42364	16’d23170	16’d42366
16‘d40960	16’d42364	16’d42371	16’d42366	16’d42366
16‘d57344	16’d42371	16’d23172	16’d42366	16’d23170

在旋转模式下分别输入遍布四个象限角度 45°、135°、225°、315°，理论值应该为 ±0.70711，由于输出结果采用 16 位有符号的二进制数来表示范围 $[-1, 1]$，所以正数理论值为 0.70711×32768＝16’d23170。负数的理论值只需要将正数理论值的二进制数取反加 1 即可得到 16’d42366。由表 4 可知最大的误差为 16’d5，得到实际结果的最大误差为 5×0.000031＝0.000155，误差精度约为 0.015%，完全满足电机驱动等工程应用要求。

向量模式下，支持输入满范围 $[-(2^{15}-1),$

$2^{15}-1$]的初始向量坐标(x_0,y_0)来计算 arctan $\frac{y_0}{x_0}$,输出结果为[0,$2^{16}-1$],代表 0～360°。实际计算结果和理论值比较如表 5 所示:

表 5 向量模式下计算结果与理论值的比较

x_0	y_0	输出角度	角度理论值	误差值(°)
16'd23170	16'd23170	16'd8191	16'd8191	<0.0054
16'd42366	16'd23170	16'd24576	16'd24576	<0.0054
16'd42366	16'd42366	16'd40958	16'd40959	0.0054
16'd23170	16'd42366	16'd57344	16'd57343	0.0054

表 6 文献[3]中向量模式结果与理论值的比较

x_0	y_0	输出角度	角度理论值	误差值(°)
16'h4e20	16'h7530	16'd922474	16'd922582	0.0065
16'hb1e0	16'h7530	16'd2026646	16'd2026537	0.0066
16'hb1e0	16'h8ad0	16'd3871594	16'd3871703	0.0065
16 h4e20	16'h8ad0	16'd4975766	16'd4975658	0.0065

处理器输出的结果范围在十进制数(0,65535)之内,对应输出角度 0～360°,所以经过换算得到最小的单位角度为 0.0054°。经过实际的计算得知计算结果与理论值偏差很小,误差值最大仅为 0.0054°,相较于表 6 文献[3]中向量模式下计算最大误差值 0.0066°,误差减少了约 18.18%,从而进一步提高了计算结果的准确性。

4 结 论

基于传统的 CORDIC 算法原理,采用 Verlog 硬件描述语言完成 MATH 处理器的电路设计及仿真验证,此电路主要针对传统 CORDIC 算法迭代周期长、运算精度低、输入角度和坐标范围小等局限性进行优化,通过角度预处理以及区间映射来扩大输入旋转角度的范围,通过坐标判断、镜像迭代将输入向量坐标扩展到整个圆周,再通过合并迭代结构进行迭代计算以此缩短迭代周期,提高运算速度,最后通过角度补偿及区间换算进行还原输出。在硬件实现上,本设计具有高精度低延时,输入坐标和角度范围广等特点,因此适用于有高精度三角函数运算需求的工程应用场景。

参考文献

[1] 郑传喜,古元冬.高精度双向同步旋转 CORDIC 算法设计与实现[J].上海大学学报(自然科学版),2022,28(5):872－882.

[2] 李春娟,李沙.基于自适应旋转角度的 CORDIC 算法的设计与仿真[J].数字技术与应用,2021,39(4):113－115.

[3] 刘奥林,黄嵩人.基于 CORDIC 算法的反正切函数 IP 核的设计与优化[J].中国集成电路,2022,31(5):32－36.

[4] 仲雅莉,吴俊辉,刘炫,等.一种基于 CORDIC 算法的高精度反正切求解[J].电子技术应,2022,48(1):12－17.

[5] 王瑜.复杂探测环境下目标检测算法的优化及其在 FPGA 中的实现[D].武汉:华中科技大学,2019.

[6] 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.

[7] 程鹏.基于改进 CORDIC 算法的虚拟频谱分析仪设计与实现[D].成都:西南交通大学,2021.

[8] 汪旭兴.基于 CORDIC 算法数字下变频器的 ASIC 实现[D].北京:北方工业大学,2020.

[9] 张志明.基于 RISC-V 架构三角函数协处理器的设计[D].南京:东南大学,2021.

[10] WEN Q G, LIANG Y C, WU C G, et al. A novel iterative velocity control algorithm and its FPGA implementation based on trigonometric function[J]. Chinese Journal of Electronics, 2019, 28(2): 237－245.