多模式 CORDIC 算法结构改进与实现

刘小宁[1,2],谢宜壮[2],陈 禾[2],李炳沂[2]
(1. 航天东方红卫星有限公司,北京 100094;2. 北京理工大学嵌入式实时信息处理技术北京市重点实验室,北京 100081)

摘 要: 本文对计算反正余弦函数的 CORDIC 算法的迭代结构进行了改进,并在此基础上完成多模式 CORDIC 算法的实现. 通过重新设定初始旋转向量避免了前两级迭代,通过修改向量旋转方向的判决条件对原算法的误差进行了校正,在增加了很少资源的情况下将正余弦运算和反正余弦运算统一到同样的迭代结构中并予以实现. 实现结果表明改进后的算法反正余弦运算结果有更高的运算精度,在两种运算函数都需要的应用中能够有效减少的硬件资源占用.

关键词: 多模式; 坐标旋转数字计算; 双迭代法; 三角函数
中图分类号: TN331 **文献标识码:** A **文章编号:** 0372-2112 (2018)02-0495-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.02.032

Improvement and Implementation of Multi-mode Architecture for CORDIC Algorithm

LIU Xiao-ning[1,2], XIE Yi-zhuang[2], CHEN He[2], LI Bing-yi[2]
(1. *DFH Satellite co., Ltd., Beijing* 100094, *China*;
2. *Beijing Key Laboratory of Embedded Real-time Information Processing Technology, Beijing Institute of Technology, Beijing* 100081, *China*)

Abstract: This paper improves the iterative structure of the CORDIC algorithm, and implements a multi-mode CORDIC algorithm. We reset the initial rotation to avoid the first two stage iterations and fix the error of the original algorithm by modify the judgment condition which determine the rotation direction of the vector. In the case of increasing few hardware resources, the sine, cosine, arcsine and arccosine operations are unified to the same iterative structure and be realized. The results show that the improved algorithm has higher accuracy and the hardware resources utilization can be effectively reduced in the application that need both of the two kinds of functions.

Key words: multi-mode; coordinate rotation digital computer (CORDIC); double iteration algorithm; trigonometric functions

1 引言

坐标旋转数字计算(coordinate rotation digital computer, CORDIC)算法是 J. Volder 等人于 1959 年在美国航空控制系统的设计中提出来的[1],1971 年 J. Walther 提出了统一的 CORDIC 算法实现结构[2],该算法在随后的几十年得到不断的改进和优化[3~6]. CORDIC 算法的优势在于通过简单的移位和加减法运算就能实现包括三角函数在内的一些复杂的超越函数. 由于该算法是一种规则化的算法,它满足了硬件对算法的模块化、规则化的要求,是硬件与算法相结合的一种优化方案,因而被广泛应用于各种工程实现[6~8].

常规的 CORDIC 算法的迭代结构可以用来计算正余弦、反正切等多种超越函数,由于在计算反正余弦函数时需要对向量旋转带来的增益进行补偿,进而引入乘法和平方根运算,增加了硬件实现难度,因此常规的 CORDIC 算法不适合于反正余弦函数的运算. 文献[9]利用传统的 CORDIC 算法实现了反正弦运算,但是由于没有对旋转因子进行补偿可以计算的角度范围有限而且结果不精确. 文献[10]提出了的免缩放因子的双步旋转 CORDIC 算法,但并不适用于反正余弦函数的运算. 文献[12]中 T. Lang 给出了利用常规 CORDIC 迭代结构进行反正余弦运算的方法,但是需要引入乘法运算. 文献[13]中 C. Mazenc 等人对常规 CORDIC 算法的迭代结构进行了改进,提出了双迭法,能够用来计算反正弦和反余弦函数,但是需要进行改进和优化.

收稿日期:2015-06-09;修回日期:2015-11-20;责任编辑:蓝红杰
基金项目:国家 863 高技术研究发展计划(No. 2011AA120202);国家科技重大专项(No. D020205)

由于计算正余弦函数的常规 CORDIC 算法和计算反正余弦函数的双迭代法的结构不同,因此在两种函数都需要计算的算法实现中,计算这两种三角函数需要分别占据硬件资源,这样会造成过多的硬件资源消耗. 本文首先对文献[13]中的双迭代法进行了资源的优化和误差的校正,然后将正余弦函数和反正余弦函数运算统一到双迭代法的迭代结构中,最后将改进后的算法在 FPGA 上采用流水结构进行硬件实现. 改进后的算法具有正余弦和反正余弦两种工作模式,用户可以根据需求对算法功能进行实时切换,因此在实际应用中相比使用两个单独的 CORDIC 运算核能够有效节省硬件资源占用面积.

2 CORDIC 算法优化与改进

2.1 双迭代法

常规的 CORDIC 算法迭代结构在计算反正余弦函数时,为了补偿向量旋转带来的增益,会引入乘法和平方根运算. 平方根运算无法用简单的加法、减法或者移位操作来代替. C. Mazenc 在文献[13]中提出了双迭代法"Double Iteration Algorithm",双迭代法的主要优势在于将随着向量旋转而变化的参数由

$$t_{i+1}=t_i/\cos(\tan^{-1}2^{-i})=t_i\sqrt{1+2^{-2i}} \tag{1}$$

变成了

$$t_{i+1}=t_i/\cos^2(\tan^{-1}2^{-i})=t_i(1+2^{-2i}) \tag{2}$$

这样乘法和平方根运算就变成了加法和移位运算. 双迭代法的迭代关系如公式(3)所示.

$$\begin{cases} x_0=1,y_0=0,z_0=0,t_0=t \\ d_i=\mathrm{sign}(x_i)\ \text{if}\ y_i\leqslant t_i,\text{else-sign}(x_i) \\ \begin{bmatrix}x_{i+1}\\ y_{i+1}\end{bmatrix}=\begin{bmatrix}1 & -d_i\cdot 2^{-i}\\ d_i\cdot 2^{-i} & 1\end{bmatrix}^2\begin{bmatrix}x_i\\ y_i\end{bmatrix} \\ z_{i+1}=z_i+2\cdot d_i\cdot\tan^{-1}2^{-i} \\ t_{i+1}=t_i+t_i\cdot 2^{-2i}\end{cases} \tag{3}$$

按照公式(3)的迭代关系,输入参数的反正弦值可由 $\theta=\sin^{-1}t=z_{i+1}$ 得到,反余弦值可由反正弦值和反余弦值的转换关系 $\cos^{-1}t=\pi/2-\sin^{-1}t$ 得到. 对于双迭代法,函数的收敛域为:

$$\begin{aligned}\theta&\in\left[-2\sum_{i=0}^{\infty}\tan^{-1}2^{-i},2\sum_{i=0}^{\infty}\tan^{-1}2^{-i}\right] \\ &\approx[-3.486,3.486]\end{aligned} \tag{4}$$

因此,对于任何的 $\boldsymbol{t}\in[-1,1]$,都可通过双迭代法求得其反正余弦值.

2.2 迭代级数优化

在实际工程应用中,使用流水结构实现 CORDIC 算法时,每一级迭代都需要占据硬件资源. 本文通过研究双迭代法的向量旋转趋势,在不影响算法结果精度的前提下减少了前两级迭代,从而达到了缩减硬件资源的目的.

由于反正弦函数是奇函数,公式(3)可以做如下修改:

$$\begin{cases} x_0=1,y_0=0,z_0=0,t_0=\mathrm{abs}(t) \\ d_i=\mathrm{sign}(x_i)\quad \text{if}\ y_i\leqslant t_i\quad \text{else}-\mathrm{sign}(x_i) \\ \begin{bmatrix}x_{i+1}\\ y_{i+1}\end{bmatrix}=\begin{bmatrix}1 & -d_i\cdot 2^{-i}\\ d_i\cdot 2^{-i} & 1\end{bmatrix}^2\begin{bmatrix}x_i\\ y_i\end{bmatrix} \\ z_{i+1}=z_i+2\cdot d_i\cdot\tan^{-1}2^{-i} \\ t_{i+1}=t_i+t_i\cdot 2^{-2i}\end{cases} \tag{5}$$

反正弦函数的结果由 $\theta=\sin^{-1}t=\mathrm{sign}(t)\cdot z_{i+1}$ 计算得出. 向量(x_i,y_i)会一直在第一象限和第二象限旋转,向量旋转过程中 y_i 和 z_i 一直会是正值.

由公式(5)可以得出,初始向量(x_0,y_0)为 X 轴上的单位向量. 如图 1 所示,在第一级迭代中,d_0 的值必然为 1,向量(x_0,y_0)会逆时针旋转 $2*\tan^{-1}(2^{-0})=90°$,这样(x_1,y_1)则为 Y 轴上的一个向量. 在第二级迭代中,由于目标角所在区间为$[-90°,90°]$,因此 $d_1=-1$,(x_1,y_1)会顺时针旋转 $2*\tan^{-1}(2^{-1})$. 由此可以得知,无论目标角多大,前两级迭代的向量旋转方式是固定不变的.

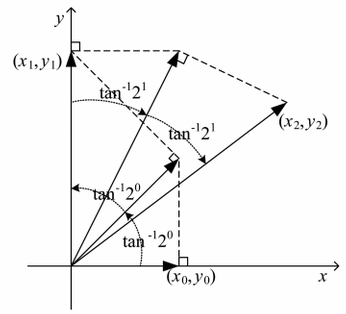


图1 双迭代法向量旋转示意图

根据以上分析,对于任意的 $t\in[-1,1]$,向量(x_0,y_0)经过前两级旋转后总会得到固定的向量(x_2,y_2). 这样,通过将初始向量变为和(x_2,y_2)同方向的单位向量,就可以将前两级迭代省略掉,从而减少了前两级迭代所占用的硬件资源.

修改后的迭代公式如下:

$$\begin{cases} x_2=0.1,y_2=0,z_0=0,t_0=\mathrm{abs}(t) \\ z_0=2\cdot\tan^{-1}2^0-2\tan^{-1}2^1 \\ \begin{bmatrix}x_{i+1}\\ y_{i+1}\end{bmatrix}=\begin{bmatrix}1 & -d_i\cdot 2^{-i}\\ d_i\cdot 2^{-i} & 1\end{bmatrix}^2\begin{bmatrix}x_i\\ y_i\end{bmatrix} \\ z_{i+1}=z_i+2\cdot d_i\cdot\tan^{-1}2^{-i} \\ t_{i+1}=t_i+t_i\cdot 2^{-2i}\end{cases} \tag{6}$$

2.3 误差校正

双迭代法通过改进原有的 CORDIC 算法给出了计

算反正弦函数的迭代结构,对于任意属于定义域的输入来说,都应该通过该迭代方法得出正确的反正弦值. 事实上,当输入参数的的绝对值接近于 1 的时候,通过该迭代方法得出的结果有可能会超出反正弦函数的值域范围[-90°,90°].

本文通过跟踪每一级迭代的向量旋转,发现当输入参数接近于 1 的时候,向量(x_i,y_i)有可能在某一级迭代中旋转至第二象限. 因此,x_i 会得到负值,z_i 会得到大于 90°的值. 本文通过公式(7)计算出有可能导致这一问题出现的最小角度.

$$z_n = z_2 + \sum_{i=2}^{n-1} 2 \cdot \tan^{-1}2^{-i}, \quad n = 3,4,5\cdots \tag{7}$$

经计算,当 $n \geqslant 8$ 时,z_n 的值会超过 90°,$z_8 = 91.7153°$. 由此得出结论,当目标角度超过 $z_7 \approx 89.9249°$时,向量(x_i,y_i)有可能在某一级迭代中旋转至第二象限,对应输入为 $t \approx 0.9999992$,即当 $t > 0.9999992$ 时,通过双迭代法迭代计算得出的反正弦值有可能与实际值发生偏离. 例如,利用公式(6)的迭代关系,当输入为 $t = 0.9999993$ 时,会得出 $\theta = 90.0696°$.

为了弥补双迭代法的这种不足,需要做如下限制:在任意一级迭代中,无论 y_i 与 t_i 的关系和 x_i 的符号如何,只要向量(x_i,y_i)旋转至第二象限,必须在下一级迭代中旋转回第一象限,公式(6)中的向量旋转判定条件 d_i 改为

$$d_i = -1 \quad \text{if}(x_i \leqslant 0)\,\text{or}(y_i > t_i)\,,\text{else} \quad +1 \tag{8}$$

利用改进后的迭代关系,当输入 $t = 0.9999993$ 时,会得到正确的反正弦值 $\theta = 89.9283°$.

2.4　统一迭代结构

J. Walther 提出的统一 CORDIC 算法实现结构中,圆周系统下的旋转模式可以用来计算正余弦函数,其迭代关系如下:

$$\begin{cases} x_0 = 1/A_n, y_0 = 0, z_0 = \theta \\ d_i = \text{sign}(z_i) \\ \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -d_i \cdot 2^{-i} \\ d_i \cdot 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \\ z_{i+1} = z_i - d_i \cdot \tan^{-1}2^{-i} \end{cases} \tag{9}$$

其中,$A_n = \prod_{i=0}^{N-1} \sqrt{1 + 2^{-2i}}$. 经过 N 级迭代,正余弦值结果通过公式(10)求得.

$$\begin{cases} \cos\theta = x_N \\ \sin\theta = y_N \end{cases} \tag{10}$$

由于反正余弦运算核正余弦运算的迭代结构不同,在实际工程应用中,当所要实现的算法中既有正余弦函数运算又有反正余弦运算时,两种运算都要占用相应的硬件资源. 如果将正余弦运算和反正余弦运算统一成同一迭代结构,只在输入和输出中作相应处理,就能用同一硬件资源实现两种不同类型的运算,这样能够有效减少算法的硬件资源占用.

本文采用改进后的双迭代法的迭代结构来实现两种函数的运算. 由于改进后的双迭代法只在第一象限内作有效旋转,利用双迭代法计算正余弦函数时,需要利用三角函数关系式将输入数据转换至[0°,90°]范围内,如表 1 所示.

表 1　正余弦函数输入象限转换

	θ 输入范围			
	$(\frac{\pi}{2},\pi]$	$(0,\frac{\pi}{2}]$	$(-\frac{\pi}{2},0]$	$[-\pi,-\frac{\pi}{2}]$
象限 q	2	1	4	3
输出结果 θ'	$\frac{\pi}{2}-\theta$	θ	$-\theta$	$\pi+\theta$

按照图 1 所示的旋转方式,采用双迭代法每一级旋转迭代后,向量的模值就会变为原来的 $1/\cos^2(\tan^{-1}2^{-i}) = 1 + 2^{-2i}$倍,因此缩放因子由

$$A_n = \prod_{i=0}^{N-1} \sqrt{1 + 2^{-2i}} \tag{11}$$

变为:

$$A_n^2 = \prod_{i=0}^{N-1} (1 + 2^{-2i}) \tag{12}$$

设输入为 t,计算正余弦函数时采用改进后的双迭代法的初始向量和初始条件为:

$$\begin{cases} x_2 = 0.8/A_n^2, y_2 = 0.6/A_n^2 \\ z_2 = 2 \cdot \tan^{-1}2^0 - 2 \cdot \tan^{-1}2^1 \\ t_2 = \text{chg}(t) \end{cases} \tag{13}$$

其中 chg (t)表示根据表 1 的三角函数关系式对输入数据 t 做角度转换.

计算反正余弦时的初始向量和初始条件为:

$$\begin{cases} x_2 = 0.8/A_n^2, y_2 = 0.6/A_n^2 \\ z_2 = 2 \cdot \tan^{-1}2^0 - 2 \cdot \tan^{-1}2^1 \\ t_2 = \text{abs}(t) \end{cases} \tag{14}$$

其中 abs (t)表示对输入 t 求绝对值.

对于正余弦运算,向量旋转方向的判定条件是:

$$d_i = -1\,,\text{if}(z_i \geqslant 0)\,,\text{else} \quad +1 \tag{15}$$

对于反正余弦运算,向量旋转方向的判定条件是:

$$d_i = -1\,,\text{if}(x_i \leqslant 0)\,\text{or}(y_i > t_i)\,,\text{else} \quad +1 \tag{16}$$

为了在算法实现时节省硬件资源,正余弦运算和反正余弦运算采用相同的迭代结构:

$$\begin{cases} \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -d_i \cdot 2^{-i} \\ d_i \cdot 2^{-i} & 1 \end{bmatrix}^2 \begin{bmatrix} x_i \\ y_i \end{bmatrix} \\ z_{i+1} = z_i + 2 \cdot d_i \cdot \tan^{-1}2^{-i} \\ t_{i+1} = t_i + t_i \cdot 2^{-2i} \end{cases} \tag{17}$$

由于迭代结构相同,两种运算可以采用共同的硬件资源来实现,能够有效减少硬件资源占用.

经过 N 级迭代,正余弦函数运算结果可以通过最后一级迭代输出的 x_N 和 y_N 经过象限恢复得到. 如表 2 所示.

表 2 正余弦函数输出象限恢复

输出正余弦值	输入参数 θ 所在象限			
	1	2	3	4
$\cos\theta$	x_N	$-x_N$	$-x_N$	x_N
$\sin\theta$	y_N	y_N	$-y_N$	$-y_N$

对于反正余弦值,可以通过公式(18)求得.

$$\begin{cases}\theta_{\text{asin}} = \sin^{-1} t = \text{sign}\ (t) \cdot z_N \\ \theta_{\text{acos}} = \cos^{-1} t = \dfrac{\pi}{2} - \text{sign}\ (t) \cdot z_N\end{cases} \tag{18}$$

3 算法实现

我们使用 VHDL 硬件编程语言对改进后的算法进行了实现,并在实际工程中加以应用. 算法在硬件结构上采用了全流水的实现方式,能够实现输入参数的连续输入和正余弦函数、反正余弦函数的实时切换. 为了节省硬件资源,我们使用定点运算代替浮点,定点运算数据路径位宽 BIT_WIDTH 和迭代级数 ITERATION_NUM 可以根据不同的精度需求进行参数化设置. 改进后的算法实现结构如图 2 所示.

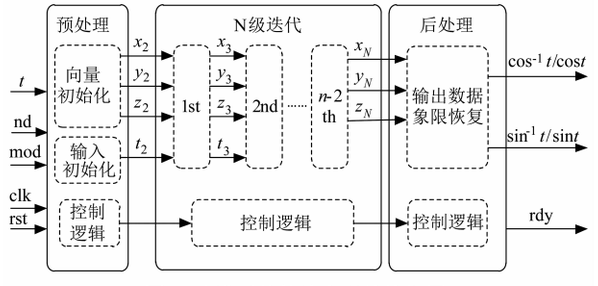


图2 改进后的算法实现结构框图

如上图所示,算法的实现主要分为三个功能模块,预处理模块、N 级迭代和后处理模块. 预处理模块根据所要实现的函数功能对输入数据进行初始化,设置初始向量的值. N 级迭代模块根据设置的 ITERATION_NUM 值,完成算法的 $2N$ 次向量旋转运算. 由于采用流水结构,每个模块都有相应的控制逻辑,传递当前一级迭代的工作模式、输入参数象限或符号信息. 后处理模块则根据传递过来的对应输入数据的象限信息或者数据符号将最后一级迭代输出的数据转换成正余弦值或者反正余弦值,并将结果寄存输出.

4 结果分析

本文对 C. Mazenc 的双迭代法和改进后的正余弦反正余弦统一结构的算法实现后的 VHDL 代码在 Xilinx 公司的 Virtex 6 系列 FPGA XC6VLX240T 上进行了综合和仿真,并将两者硬件资源占用情况、运算结果误差进行了对比,同时给出了改进后的算法在不同数据路径位宽和迭代级数下的信噪比分析结果,用以指导用户在满足信噪比要求下选择合适的参数以减少硬件资源占用.

4.1 硬件资源占用

本文将 C. Mazenc 的双迭代法和改进后的正余弦反正余弦统一结构的算法的 FPGA 硬件资源占用进行了比较,当 ITERATION_NUM = 12, BIT_WIDTH = 20 时 LUT 和 Register 占用情况如表 3 所示.

表 3 硬件资源占用对比

硬件资源占用	Registers	LUTs
原始双迭代法[13]	911	1545
改进后的算法	1040	1678
资源增加百分比	14%	9%

由上表可以看出,虽然改进后的算法比原始的双迭代法多占用 14% 的 Register 和 8% 的 LUT,但是改进后的算法能够完成正余弦和反正余弦两种工作模式的实时切换. 原始的双迭代法仅仅能够进行反正余弦运算,因此在正余弦和反正余弦两种函数运算都需要的算法中,能够有效的节省硬件资源占用.

4.2 波形仿真

本文利用 Modelsim10.1a 仿真工具对 FPGA 代码进行仿真. 正余弦运算输入数据由[$-\pi$, π]均匀采样 2000 点,反正余弦运算输入数据由[-1, 1]均匀采样 2000 点. 仿真时,连续输入 2000 点正余弦运算输入数据和 2000 点反正余弦输入数据,仿真波形如图 3 所示.

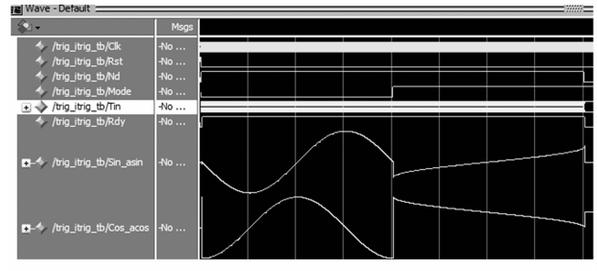


图3 改进后的算法仿真波形

由上图可以看出,改进后的算法能够正确输出正余弦和反正余弦波形,并且实现正余弦和反正余弦两种运算模式的即时切换.

4.3 误差对比

本文以 Matlab 的自带反正弦函数运算结果为标准,对算法改进前后的反正弦结果进行了误差对比. 数

据路径位宽和迭代级数分别为 BIT_WIDTH = 29, ITERATION_NUM = 20,输入参数在 0.9999992 附近进行选取,如表 4 所示.

表 4　算法改进前后反正弦结果误差对比

输入参数	输出反正弦值及误差				
	Matlab	原始双迭代法[13]		改进后的算法	
	输出值	输出值	误差	输出值	误差
0.9999990	89.9190°	89.9198°	0.0008°	89.9198°	0.0008°
0.9999991	89.9231°	89.9250°	0.0019°	89.9250°	0.0019°
0.9999992	89.9275°	90.0737°	0.1462°	89.9248°	-0.0027°
0.9999993	89.9322°	90.0696°	0.1374°	89.9283°	-0.0039°
0.9999996	89.9488°	90.0548°	0.1060°	89.9454°	-0.0034°
0.9999998	89.9638°	90.0413°	0.0775°	89.9597°	-0.0041°
0.9999999	89.9744°	89.9702°	-0.0042°	89.9702°	-0.0042°

由表 4 可以看出,改进后的算法反正弦运算结果都在[0°,90°]范围内,并且相比改进之前有更小的误差.

4.4　信噪比分析

由于 CORDIC 算法是对所求函数的有限次逼近,数据路径位宽和迭代级数都会引入相应的量化噪声[5].本文针对不同数据路径位宽和迭代级数的算法运算结果进行了信噪比(SNR,Signal to Noise Ratio)分析,便于用户在满足需求情况下选择相应的参数,减少硬件资源占用. 对于该算法实现的正余弦运算和反正余弦运算的 SNR,按波形仿真时同样的方式各在定义域内均匀采取 2000 点数据. SNR 计算公式如下:

$$\mathrm{SNR} = 10 \cdot \lg_{10} \frac{\sum_{i=1}^{n} S_i^2}{\sum_{i=1}^{n} (R_i - S_i)^2} (\mathrm{dB}) \qquad (19)$$

表 5　改进后算法不同参数下正弦运算的 SNR(dB)

BIT_WIDTH	ITRATION_NUM						
	6	8	10	12	14	16	18
12	40.5	46.6	46.6	46.6	46.6	46.6	46.6
14	40.9	52.0	56.7	56.7	56.7	56.7	56.7
16	40.9	52.8	63.8	68.3	68.3	68.3	68.3
18	40.9	52.9	64.9	75.9	80.4	80.4	80.4
20	40.9	52.9	64.9	76.9	87.4	91.4	91.4
22	40.9	52.9	64.9	77.0	88.9	99.4	102.9
24	40.9	52.9	64.9	77.0	89.0	100.7	111.2
26	40.9	52.9	64.9	77.0	89.0	100.9	113.1

从表 5、表 6 及图 4、图 5 可知,当数据路径位宽固定时,两种函数的 SNR 随着迭代级数的增加而升高,迭代级数增加到一定程度时 SNR 趋于恒定值. 同样,当迭代级数固定时,SNR 会随数据路径位宽的增加而升高,当数据路径位宽增加到一定程度时,SNR 趋于恒定值. 虽然是同样的迭代结构,由于初始向量和向量旋转判决条件不同,两种函数的信噪比也有所不同,但随着数据路径位宽和迭代级数增加而增加的变化趋势是一致的.

表 6　改进后算法不同参数下反正弦运算的 SNR(dB)

BIT_WIDTH	ITRATION_NUM						
	6	8	10	12	14	16	18
12	36.7	41.1	40.9	40.9	40.9	40.9	40.9
14	37.6	48.5	52.5	52.2	52.2	52.2	52.2
16	37.6	49.6	60.6	64.1	64.8	64.8	64.8
18	37.6	49.6	61.5	71.4	74.4	74.3	74.3
20	37.6	49.6	61.7	73.6	83.8	85.1	86.9
22	37.6	49.6	61.7	73.7	85.5	94.5	96.8
24	37.6	49.6	61.7	73.7	85.8	97.6	107.0
26	37.6	49.6	61.7	73.7	85.8	97.9	109.6

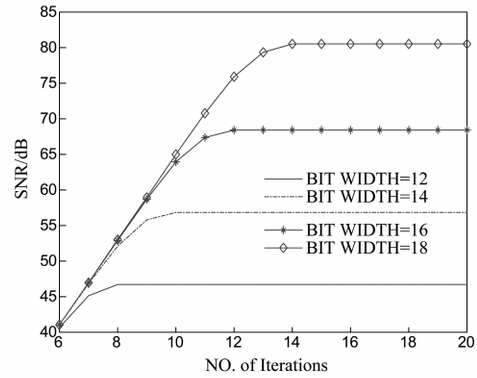


图4　不同参数下正弦运算的SNR曲线

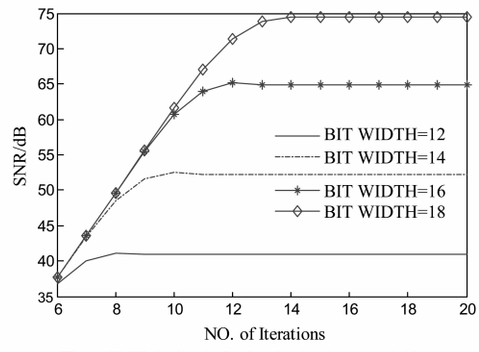


图5　不同参数下反正弦运算的SNR曲线

5 结论

本文对计算反正余弦函数的双迭代 CORDIC 算法进行了资源的优化和误差的校正,并且将正余弦函数运算和反正余弦函数运算用统一的迭代结构予以实现并给出了信噪比分析结果.改进后的算法能够对两种函数进行流水计算并可以对工作模式进行实时切换,在两种函数都需要的算法实现中,能够有效节省硬件资源占用.

参考文献

[1] Volder J E. The CORDIC trigonometric computing technique [J]. IRE Transactions on Electronic Computers, 1959,8(3):330 – 334.

[2] Walther J S. A unified algorithm for elementary functions [A]. Proceedings of Spring Joint Computer Conference [C]. New York:AFIPS,1982. 379 – 385.

[3] Li Tao, Han Yueqiu. Trigonometric function generator based on pipelined CORDIC[J]. Journal of Systems Engineering and Electronics,2000,22(4):85 – 87.

[4] Andraka R. A survey of CORDIC algorithms for FPGA based computer[A]. Proceedings of the 1998 ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays[C]. California:ACM,1998. 191 – 200.

[5] Hu H Y. The quantization effects of the CORDIC algorithm [J]. IEEE Trans on Signal Processing, 1992, 40: 834 – 844.

[6] Takagi N, Asada T. Redundant CORDIC methods with a constant scale factor for sine and cosine computation[J]. IEEE Trans on Computers,1991,40:989 – 995.

[7] 张晓彤,辛茹,王沁,李涵. 基于改进混合式 CORDIC 算法的直接数字频率合成器设计[J]. 电子学报,2008,36(6):1144 – 1148.
ZHANG Xiao-tong, XIN Ru, WANG Qin, LI Han. Design of direct digital frequency synthesizer based on improved hybrid CORDIC algorithm[J]. Acta Electronica Sinica, 2008,36(6):1144 – 1148. (in Chinese)

[8] 张晓帆,李广军. 基于低硬件复杂度、高速 CORDIC 的 SVD 模块设计与实现[J]. 电子学报,2015,43(4):738 – 742.
ZHANG Xiao-fan, LI Guang-jun. The design and implementation of SVD module with reduced hardware complexity and high-speed CORDIC processor[J]. Acta Electronica Sinica,2015,43(4):738 – 742. (in Chinese)

[9] 谢珊英,齐伟民,蔡晓宁. 基于 FPGA 的反正弦函数的实现[J]. 电子器件,2010,33(3):344 – 347.
XIE Shan-ying, QI Wei-min, CAI Xiao-ning. Implementation of arcsine Ffnction on FPGA[J]. Chinese Journal of Electron Devices,2010,33(3):344 – 347. (in Chinese)

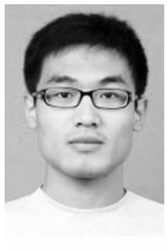
[10] 徐成,秦云川,李肯立,戚芳芳. 免缩放因子双步旋转 CORDIC 算法[J]. 电子学报,2014,42(7):1441 – 1445.
XU Cheng, QIN Yun-chuan, LI Ken-li, QI Fang-fang. Double-step scaling free CORDIC[J]. Acta Electronica Sinica,2015,43(4):738 – 742. (in Chinese)

[11] Aggarwal S, Meher P K. Reconfigurable CORDIC architectures for multi-mode and multi-trajectory operations [A]. Proceedings of the 2014 IEEE International Symposium on Circuits and Systems [C]. Melbourne: IEEE, 2014. 2490 – 2494.

[12] Lang T, Antelo E. CORDIC-based computation of ArcCos and ArcSin[A]. Proceedings of the 1997 IEEE International Conference on Application Specific Systems Architectures and Processors [C]. Zurich: IEEE, 1997. 132 – 143.

[13] Mazenc C, Merrheim X, Muller J M. Computing functions arccos and arcsin using CORDIC[J]. IEEE Transactions on Computers,1993,42:118 – 122.

作者简介



刘小宁 男,1987 年生于山东省德州市.北京理工大学信息与电子学院博士研究生,现就职于航天东方红卫星有限公司.研究方向为星上 SAR 实时成像处理与系统架构设计.
E-mail:liuxiaoning@ bit. edu. cn

谢宜壮(通信作者) 男,1980 年生于吉林省长春市.北京理工大学信息与电子学院讲师,研究方向为星上实时信息处理技术与系统架构设计.
E-mail:xyz551_bit@ bit. edu. cn