

基于CORDIC改进算法的反正切函数 在FPGA中的实现

刘小会¹, 许 蕾², 刘海颖², 王惠南¹

(1. 南京航空航天大学 航天学院, 江苏 南京 210006;

2. 南京航空航天大学 高新技术研究院, 江苏 南京 210006)

摘 要: 针对基于FPGA的分布式导航系统中涉及大量的三角函数运算, 而传统的查找表或差值法计算, 在精度、运算速度方面不能兼得, 且占用资源多, 文中提出了基于CORDIC算法的反正切函数计算的改进方法与流水线结构的实现方法, 使用VHDL硬件描述语言进行编程实现, 在Quartus II 9.0中对算法进行功能仿真, 最后通过Altera公司的FPGA Cyclone II系列芯片进行了具体验证。验证结果表明, 针对累加器中因截尾而产生的误差所作的算法改进, 显著地提高了算法精度, 而且运算速度快。

关键词: CORDIC算法; 反正切函数; VHDL; FPGA芯片; 截尾误差

中图分类号: TP391.9

文献标识码: A

文章编号: 1673-629X(2013)11-0103-05

doi: 10.3969/j.issn.1673-629X.2013.11.026

Realization of Arc-tangent Function Based on Improved CORDIC Algorithm in FPGA

LIU Xiao-hui¹, XU Lei², LIU Hai-ying², WANG Hui-nan¹

(1. College of Astronautic, Nanjing University of Aeronautics and Astronautics, Nanjing 210006, China;

2. Academy of Frontier Science, Nanjing University of Aeronautics and Astronautics, Nanjing 210006, China)

Abstract: In the light of a large number of trigonometric function calculations in the distributed navigation systems based on FPGA, while with the traditional looking-up-table or differential methods, the calculation accuracy and speed can not be got at the same time, taking up more resources, present the improved measures and pipeline structure of arc-tangent function based on CORDIC algorithm, use VHDL to program, and by using the Quartus II 9.0 the function simulation can be got, finally on the Altera FPGA chip the algorithm is tested. The results show that, truncation error generated by the accumulator are reduced significantly, and the algorithm accuracy is improved, the computing speed is very fast.

Key words: CORDIC algorithm; arc-tangent function; VHDL; FPGA chip; truncation error

0 引 言

基于惯性网络的分布式导航系统可以为运动载体(如飞机、舰船、航天器等)提供惯性数据和其他测量信息, 通过导航算法和信息融合算法, 实现导航定位、姿态确定、运动控制、惯性对准等诸多功能^[1]。文中采用FPGA进行分布式导航解算与处理, 针对其中涉及大量的三角函数运算, 提出了基于CORDIC算法的反正切函数计算的改进措施, 相对于传统的查找表或差值法计算, 以进一步提高算法精度, 并可以提高导航算法解算速度。

CORDIC (Coordinate Rotational Digital Computer, 坐标旋转计算机) 算法最早是在美国航空控制系统的设计中提出来的, 其基本思想为用一系列与运算基数相关的角度不断逼近所需旋转的角度, 本质上它是一个数值计算逼近的方法^[2]。计算基数决定这些固定的角度, 运算只有移位和加减。在传统的硬件算法设计中, 乘、除等基本数学运算是一种既耗时又占用面积大的运算, 甚至有时难以实现。CORDIC算法的提出正是为了解决这些问题的。它从算法本身出发, 将复杂的算法分解成一系列在硬件中容易实现的基本算法,

收稿日期: 2013-01-17

修回日期: 2013-04-23

网络出版时间: 2013-07-24

基金项目: 江苏省自然科学基金(KB2011729)

作者简介: 刘小会(1988-), 女, 硕士研究生, 研究方向为导航与控制; 王惠南, 教授, 研究方向为航天器空间交会状态测控技术。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20130724.0944.001.html>

如加法、移位等,从而使得这些算法在硬件上可以得到很好的实现。由于该算法是一种规则化的算法,它可以满足硬件对算法的模块化、规则化的要求,因此 CORDIC 算法是可以充分发挥硬件的优势,利用硬件的资源,从而实现硬件与算法相结合的一种优化方案。

CORDIC 算法包含圆周系统、线性系统、双曲系统三种旋转系统,每种系统又分别具有两种运算模式,即旋转模式和向量模式。文中对 CORDIC 算法的圆周系统的向量模式进行讨论,研究 CORDIC 算法中的反正切函数。

CORDIC 的实现方法分为迭代结构和流水线结构,在实际应用中,可以根据目标需求,在速度和资源之间进行折中选择。对于迭代结构,必须有一个状态机来进行跟踪迭代过程,进而控制移位器深度,并查找表的地址及决定符号因子,这直接导致了实际运行效率低下^[3]。对于流水线结构,能够在执行进程的同时输入数据,进而加快程序的运行。迭代结构虽然可以节省硬件资源,并能实现算法精度可调,但是速度慢。而流水线结构,却可以显著地提高系统速度,每增加一次迭代精度就可以提高一位,理论上可以通过增加迭代次数无限制提高精度,但是由于精度受流水线级数的限制,所以要提高精度势必要增加流水线级数。从当前 FPGA 的发展趋势来看,芯片内的门资源相对富裕受限制较少,所以对流水线 CORDIC 的实现规模约束很小。对于所讨论的向量模式,文中采用的是流水线结构,硬件采用 VHDL 进行描述,为了提高计算精度,使用了 22 次迭代运算^[4-7]。

1 CORDIC 原理

设矢量 $(x_i, y_i) = (r \cos \alpha, r \sin \alpha)$, 现将其旋转 θ 角得到新矢量 (x_j, y_j) , 则:

$$\begin{cases} x_j = r \cos(\alpha + \theta) = x_i \cos \theta - y_i \sin \theta \\ y_j = r \sin(\alpha + \theta) = y_i \sin \theta + x_i \cos \theta \end{cases} \quad (1)$$

写成矩阵的形式就是:

$$\begin{bmatrix} x_j \\ y_j \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (2)$$

上式为矢量旋转变换的通用公式。

为了旋转一个角度 θ , 可以用一个迭代的过程。将 θ 分解成若干个微旋转, 第 n 次旋转角度为 θ_n , 则有:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \cos \theta_n \cdot \begin{bmatrix} 1 & -\tan \theta_n \\ \tan \theta_n & 1 \end{bmatrix} \cdot \begin{bmatrix} x_n \\ y_n \end{bmatrix} \quad (3)$$

将 $\cos \theta_n$ 提出之后, 乘法的次数由 4 次减为 3 次, 进一步限制 $\tan \theta_n = \pm 2^{-n}$, 则可以将 $\tan \theta_n$ 乘项的乘法操作变为移位操作。

$$\theta_n = s_n \cdot \arctan 2^{-n}, s_n = \{-1, +1\} \quad (4)$$

将(4)式带入(3)式可得:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & -s_n 2^{-n} \\ s_n 2^{-n} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_n \\ y_n \end{bmatrix} \quad (5)$$

除了 $\cos \theta_n$ 外, CORDIC 算法只需要简单的移位和相加操作即可完成。考虑到

$$\cos \theta_n = \cos(s_n \cdot \arctan 2^{-n}) = \frac{1}{\sqrt{1 + 2^{-2n}}} \quad (6)$$

经过无数次迭代后, $\cos \theta_n$ 变成了一个常数:

$$\frac{1}{P} = \prod_{n=0}^{\infty} \frac{1}{\sqrt{1 + 2^{-2n}}} \approx 0.607253 \quad (7)$$

P 称为 CORDIC 算法的旋转增益。实际的算法不可能做无穷迭代, 因此实际的增益与迭代的次数有关:

$$P = \prod_n \sqrt{1 + 2^{-2n}}$$

当 N 的次数逐渐增大时, P 就会不断逼近 1.647。

由于 N 不可能去无穷大, 因此存在误差。由于 P 是一个常数, 在迭代过程中可以忽略 $\cos \theta_n$ 项, 迭代的最后再将其乘入。这样迭代式就变为:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \cos \theta_n \cdot \begin{bmatrix} 1 & -s_n 2^{-n} \\ s_n 2^{-n} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_n \\ y_n \end{bmatrix} \quad (8)$$

上面的 s_n 可以取 +1 和 -1, 它的取值直接关系到 CORDIC 算法的两种不同模式(旋转模式 rotation 和矢量模式 vectoring)。

不管是旋转模式还是矢量模式, 都要引入一个新变量 z 。

整理得以下三个迭代方程:

$$\begin{cases} x_{n+1} = x_n - s_n 2^{-n} y_n \\ y_{n+1} = y_n + s_n 2^{-n} x_n \\ z_{n+1} = z_n - s_n \arctan 2^{-n} \end{cases} \quad (9)$$

在矢量模式下:

$$s_n = \begin{cases} -1, y_n \geq 0 \\ +1, y_n < 0 \end{cases} \quad (10)$$

z_n 表示 $\arctan \frac{y_0}{x_0}$, 表示最后需要旋转的角度。

文中采用流水线结构, 该结构中, 每一个移位器都是固定的深度, 而且旋转角度集的各个值作为常数直接连接到了累加器上, 不需要存储空间和读取时间。整个 CORDIC 简化为加法器的直接连接, 大多数器件的每个单元中都有寄存器, 便于采用流水线技术。CORDIC 流水线结构如图 1 所示。

2 基于 CORDIC 原理的反正切函数优化实现

浮点反正切函数的硬件结构主要包含三部分, 如

图 2 所示。

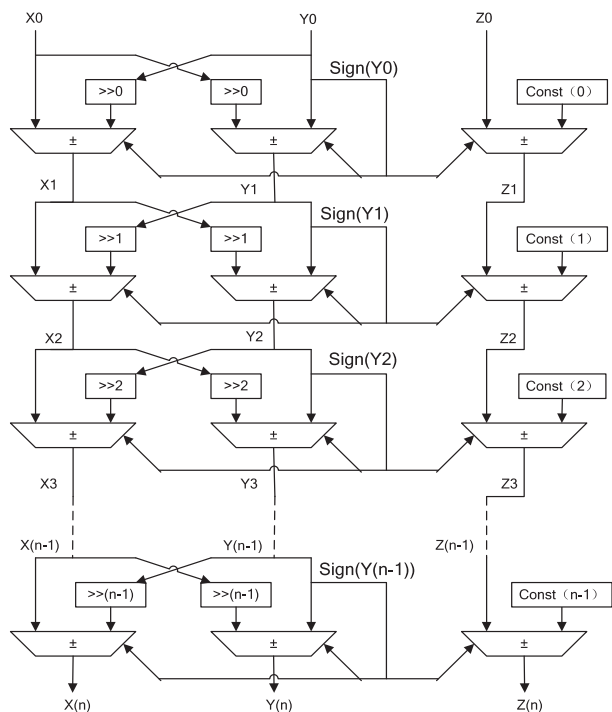


图 1 CORDIC 算法实现反正切函数流水线结构



图 2 整体框图

由 CORDIC 原理^[4,8-11]可知 θ 的取值范围为 $[-99.88^\circ, 99.88^\circ]$ ，而一般要求角度覆盖 $[-\pi, \pi]$ ，所以需要进行预处理^[12]。当角度小于 $-\pi/2$ 或者大于 $\pi/2$ 时，要想办法使之落于 $[-\pi, \pi]$ 。处理方法：将整个平面划分为四个象限。当向量位于 $(+, +)$, $(+, -)$ 区域时，输入的 x, y 值不变；当向量位于 $(-, +)$, $(-, -)$ 区域时令 $x' = -x, y' = -y$ 。执行完 CORDIC 算法后，要得到最终的角度值，需要根据事先设置的标志位的值对角度进行后处理。

由 CORDIC 算法基本迭代公式可知，CORDIC 算法主要通过移位运算和加减运算对超越函数进行计算，因为整数的移位和加减对于硬件电路来说实现相对容易，并且运算速度可以做得很快；而浮点格式的加减运算则非常复杂，运算速度慢，一般不采用浮点格式的数据类型来设计硬件 CORDIC 内核。由于 VHDL 在综合时只能对定点数进行运算，所以必须对数值进行扩大，这样必然导致结果也扩大，所以要进行后处理，乘以相应的换算因子，使结果变为原始的数值。在硬件实现上，需要对输入值 x, y 的指数位进行比较，以指数大的那个值作为基准，将指数较小的那个值尾数右移相应的位数，这样就可以实现从 IEEE-754 浮点数据类型到定点整型数据类型的转换。

文中在上面硬件实现基础上进一步改进以减小精度损失。设计的反正切函数硬件模块输入为 IEEE-754 单精度浮点数据，而模块内部迭代使用的是定点整型数据，因此需要转换。矢量 (x, y) 在平面坐标系中的角度为 $\arctan(y/x)$ ，它只与 y, x 的比值有关，而与矢量的实际长度无关。在迭代过程中向量的长度由于畸变因子的存在不断地发生改变，文中需要解决的是角度问题，在迭代过程中由于角度和向量的长度无关所以可以不考虑畸变因子的影响。截尾误差很小，但是它会给计算序列带来可累积的偏差效应。为了减少因截尾而产生的角度误差，文中对输入的 IEEE-754 单精度浮点数作如下处理：IEEE-754 单精度浮点数据格式包括 1 位符号位、8 位指数位和 23 位小数位。并且在指数位和小数位之间还包含了一位隐含的“1”，所以共有 24 位小数位。尾数通常表示的值在大于等于 1.0 到 2.0 之间。尾数的最高位是隐含的“1”，正好是在二进制小数点左边的第一位，而余下的 23 位在小数点右边。文中讨论的定点整型数据的位宽为 32 位，浮点数的表示范围为 $-32\ 768 \sim 32\ 767$ ，由于 1 和 327 67 浮点表示数的指数位差值为 14，因此在进行数据处理时会以 14 为分界点。先对两输入值进行预处理，再将浮点数的尾数放到长度为 38 位的变量中的第 22 ~ 0 位，然后将该变量的第 23 位置为“1”（即为隐含的“1”）。当两输入值的指数差值 n 大于 14 时，将指数值大的输入值经过处理后左移 14 位放到位数为 38 位变量中的第 37 ~ 14 位，将另一变量经预处理后右移 $n - 14$ 位，同样放入位数为 38 位的变量中；当两输入值的指数差值 n 不大于 14 时，将指数值大的输入值经预处理后左移 14 位放入位数为 38 位变量中的第 37 ~ 14 位，将另一变量经预处理后左移 $14 - n$ 位，同样放入位数为 38 位的变量中。上述移位过程中，未涉及的位则置“0”。如果输入值之中有一个为 0，那么将另一输入值处理为 38 位全为“1”的变量；如果输入值均为 0，那么输入值均处理为 38 位全为“1”的变量。最后，把 38 位变量中的第 37 ~ 9 位中的值赋给输出量的第 28 ~ 0 位，输出量的第 31 ~ 29 位置“0”（输出量为 32 位），然后根据输入值的正负，相应地给输出量取正值或负值。图 3 所示为对 IEEE-754 单精度浮点数的处理过程。

3 改进算法验证与性能分析

在使用 quartus 对模块进行功能仿真时，为了全面地验证改进后的算法是否有效地提高了精度，文中所取的 (x, y) 坐标点均匀地分布于坐标系内，分别于半径 1, 10, 100, 10 000 附近取点，共 64 个点，位于图 4 中小圆所圈出的点附近。

模块在 Altera 公司的 FPGA 器件中进行了硬件验

证,采用的综合工具是 Quartus 9.0,选用的芯片是 Cyclone 系列 EP2C35F484C6。综合以后,一共需要5 150个逻辑单元,约占用 16% 的芯片资源。经过验证,四个象限的反正切函数算法经过改进后精度都得到了显著提高。

由于篇幅有限,只列出了一部分综合时序分析报告以及第一象限处理结果。由图 5 的时序报告可以看出由于移位次数的增加,运算延时变长。

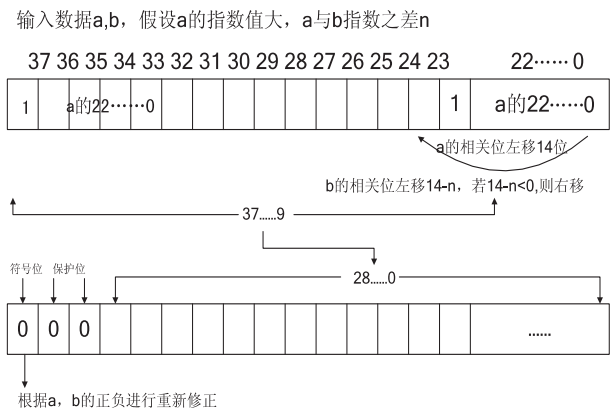


图 3 输入 IEEE-754 单精度浮点数的输入过程

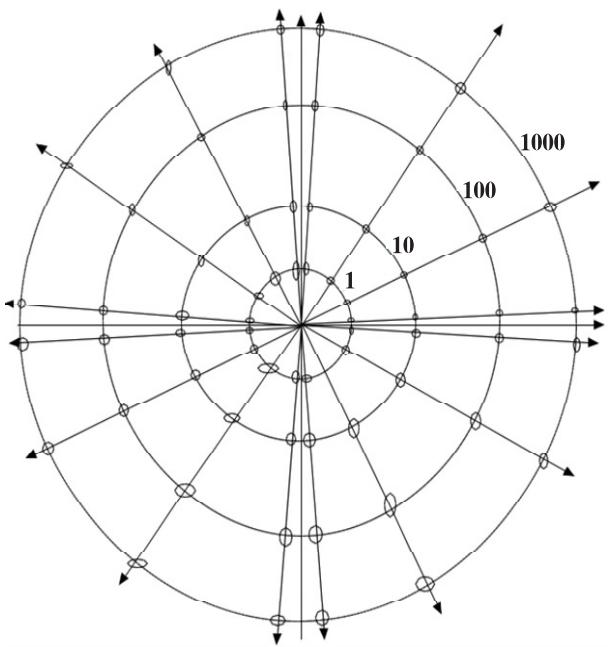


图 4 取点分布示意图

表 1 中为改进后第一象限所取点的仿真结果、未改进时的仿真结果、理论值,以及相应的误差计算。

	160.0 ns	170.0 ns	180.0 ns	190.0 ns	200.0 ns	210.0 ns	220.0 ns	230.0 ns	240.0 ns
0	clock								
1	data_a	00000000	C6066000	00000000	BF000000	00000000	C0A00000	00000000	C2480000
34	data_b	00000000	C59C4000	00000000	BF5C28F6	00000000	C109999A	00000000	C2AC0000
67	result		00000000		3E30A120	00000000	3D432E80	00000000	3A4C6000

图 5 时序仿真

表 1 仿真结果及理论分析表

x	y	Arctan(y/x) 改进后	改进误差	Arctan(y/x) 未改进	未改进误差	理论值
1	0.000 1	0.000 200 7	100.7%	0.172 489 6	error	0.000 1
10	0.001	0.000 099 6	-0.4%	0.047 651 7	error	0.000 1
100	0.01	0.000 099 6	-0.4%	0.000 779 6	error	0.000 1
10 000	1	0.000 099 6	-0.4%	0.000 072 9	-27.1%	0.000 1
0.86	0.5	0.526 628	1.899e-6	0.172 489 6	error	0.526 627
8.6	5	0.526 628	1.899e-6	0.442 305	-16.01%	0.526 627
86	50	0.526 628	1.899e-6	0.520 41	-1.18%	0.526 627
8 600	5 000	0.526 628	1.899e-6	0.526 633 7	1.27e-5	0.526 627
0.5	0.86	1.044 167 9	-1.05e-6	1.570 796 4	error	1.044 169 0
5	8.6	1.044 167 9	-1.05e-6	1.003 515 7	-3.89%	1.044 169 0
50	86	1.044 167 9	-1.05e-6	1.034 762 8	-0.9%	1.044 169 0
5 000	8 600	1.044 167 9	-1.05e-6	1.044 038 2	-1.25%	1.044 169 0
0.000 1	1	1.570 696 3	5.73e-4	1.570 796 4	6.37e-4	1.569 796 3
0.001	10	1.570 696 3	5.73e-4	1.695 796 4	8.03%	1.569 796 3
0.01	100	1.570 696 3	5.73e-4	1.570 796 4	6.37e-4	1.569 796 3
1	10 000	1.570 696 3	5.73e-4	1.570 721 1	5.89e-4	1.569 796 3

由表中可以看出改进后,反正切函数精度大大地提高了,且反正切的值只与 x,y 的比值有关,而与向量的长度无关。而改进之前,反正切的值是与向量的长度有关的,向量长度越小,误差值越大,当 x,y 值小于 1 时,甚至发生严重的错误;当 x,y 为小数时,由于浮点转整形的截尾会产生误差,所以反正切函数值误差也会很大,而改进后这种误差会被大大地减弱。由于算法的迭代次数有限,所以对于极小角度,得到的反正切值误差较大,如果需提高小角度的精度,可以增加迭代次数,这样造成的结果是占用的资源变多,并且运行的周期变长,在实际应用中需要在精度和速度两者之间进行折中。

4 结束语

文中提出了 CORDIC 算法的改进方法,给出了浮点反正切函数实现的硬件结构图,并在 Quartus ii 9.0 环境下,在 Altera 公司的 FPGA Cyclone 系列 EP2C35F484C6 芯片上进行验证,其精度得到了大大的提高,运行速度快,硬件资源占用少。文中提出的改进方法已经在某型分布式导航系统和光电吊舱的实现过程中得到应用,该算法也适用于其他基于 FGPA 的系统实现中。

参考文献:

[1] Heredia G, Ollero A. Virtual sensor for failure detection, identification and recovery in the transition phase of a morphing aircraft[J]. Sensors, 2010, 10(3): 2188-2201.

[2] Volder J E. The cordic trigonometric computing technique[J]. IRE Trans on Electronic Computers, 1959, EC-8(3): 330-334.

[3] Vachhani L, Sridharan K, Meher P K. Efficient CORDIC algorithms and architectures for low area and high throughput implementation[J]. IEEE Transactions on Circuits and Systems-II: Express Briefs, 2009, 56(1): 61-65.

[4] 徐光辉, 程东旭, 黄如. 基于 FPGA 的嵌入式开发与应用[M]. 北京: 电子工业出版社, 2006.

[5] 张建斌, 梁芳, 刘乃安. 一种改进型 CORDIC 算法的 FPGA 实现[J]. 微电子学与计算机, 2010, 27(11): 181-184.

[6] 李全, 李晓环, 陈石平. 基于 CORDIC 算法高精度浮点超越函数的 FPGA 实现[J]. 电子技术应用, 2009(5): 166-170.

[7] 段文伟, 于龙洋, 李署坚. 一种改进的 CORDIC 算法及其 FPGA 实现[J]. 微电子学与计算机, 2012, 29(2): 95-98.

[8] 李美俊, 李光明. 基于嵌入式的 CORDIC 算法的改进及实现[J]. 微电子学与计算机, 2012, 29(2): 142-145.

[9] 骆艳卜, 张会生, 张斌, 等. 一种 CORDIC 算法的 FPGA 实现[J]. 计算机仿真, 2009, 26(9): 305-307.

[10] 张天瑜. 基于旋转模式的改进型 CORDIC 算法研究[J]. 微电子学与计算机, 2010, 27(3): 93-97.

[11] 辛艳, 李环. 改进型 CORDIC 算法及 FPGA 的实现[J]. 沈阳理工大学学报, 2010, 29(5): 34-37.

[12] 张俊涛, 王红仓. 基于 FPGA 的 CORDIC 算法通用 IP 核设计[J]. 微计算机信息, 2008, 24(7-3): 238-240.

+++++

(上接第 102 页)

framework[J]. Australasian journal of educational technology, 2009, 25(4): 544-558.

[5] 史忠植. 知识发现[M]. 北京: 清华大学出版社, 2002.

[6] Han Jiawei, Kamber M. Data mining: concepts and techniques[M]. San Francisco: Morgan Kaufmann Publishers, 2000.

[7] Grabmeier J, Rudolph A. Techniques of cluster algorithms in data mining[J]. Data mining and knowledge discovery, 2002, 6(4): 303-360.

[8] Jain A K, Murty M N, Flynn P J. Data clustering: a review[J]. ACM computing surveys, 1999, 31(3): 264-323.

[9] MacQueen J. Some methods for classification and analysis of multivariate observations[C]//Proc of the 5th symposium on mathematical statistics and probability. Berkeley: [s. n.], 1967: 281-297.

[10] Kaufman J, Rousseeuw P J. Finding groups in data: an intro-

duction to cluster analysis[M]. New York: John Wiley & Sons, 1990.

[11] 马帅, 王腾蛟, 唐世渭, 等. 一种基于参考点和密度的快速聚类算法[J]. 软件学报, 2003, 14(6): 1089-1095.

[12] Gospodnetic O, Hatcher E. Lucene in action2[M]. Stamford: Manning Publications Co, 2010.

[13] Salton G, Clement T Y. On the construction of effective vocabularies for information retrieval[EB/OL]. 2011-02-04. <http://dl.acm.org/citation.cfm?id=951766>.

[14] Salton G, Wong A, Yang C. A vector space model for automatic indexing[J]. Communications of the ACM, 1975, 18(11): 613-620.

[15] 刘远超, 王晓龙, 徐志明, 等. 文本聚类综述[J]. 中文信息学报, 2006, 20(3): 55-62.

基于CORDIC改进算法的反正切函数在FPGA中的实现

作者：

[刘小会](#)，[许蕾](#)，[刘海颖](#)，[王惠南](#)，[LIU Xiao-hui](#)，[XU Lei](#)，[LIU Hai-ying](#)，[WANG Hui-nan](#)

作者单位：

[刘小会, 王惠南, LIU Xiao-hui, WANG Hui-nan\(南京航空航天大学 航天学院, 江苏 南京, 210006\)](#)，[许蕾, 刘海颖, XU Lei, LIU Hai-ying\(南京航空航天大学 高新技术研究院, 江苏 南京, 210006\)](#)

刊名：

[计算机技术与发展](#)

英文刊名：

ISTIC

[Computer Technology and Development](#)

年，卷(期)：

[2013\(11\)](#)

本文链接：http://d.g.wanfangdata.com.cn/Periodical_wjz201311027.aspx