免缩放因子双步旋转 CORDIC 算法

徐 成,秦云川,李肯立,戚芳芳
(湖南大学信息科学与工程学院,湖南长沙 410082)

摘 要: 集成电路设计中经常使用 CORDIC 算法实现高效的向量旋转操作.当前对该算法的研究热点集中在减少该算法的迭代次数、扩展其收敛范围以及降低缩放因子补偿操作的代价等问题上.本文提出免缩放因子的双步旋转 CORDIC 算法使用双步旋转策略,减少了免缩放因子 CORDIC 算法的迭代次数,将收敛区间扩展到了整个圆周区间.实验结果表明,该算法保持高计算精度的同时减少了迭代次数和面积消耗.

关键词: 双步旋转; CORDIC 算法; 区间折叠
中图分类号: TN492 **文献标识码:** A **文章编号:** 0372-2112 (2014)07-1441-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.07.031

Double-Step Scaling Free CORDIC

XU Cheng, QIN Yun-chuan, LI Ken-li, QI Fang-fang
(*College of Information Science and Engineering, Hunan University, Changsha, Hunan* 410082, *China*)

Abstract: CORDIC (Coordinate Rotation Digital Computer) can be an efficient vector rotation algorithm in the design of integrated circuit. Many researches focus on the reduction of iteration times, and to expand the scope of the convergence and reduce the cost of scaling factor compensation. This paper presents a double-step scaling-free CORDIC algorithm which uses two-step rotation strategies. The algorithm reduces the iteration times, and extends the convergence range to the entire circumference. The experiments show that the algorithm has excellent performance in terms of calculation accuracy, iteration times, and the area consumption.

Key words: double-step rotation; CORDIC; domain folding

1 引言

现代信号处理,图像处理以及现代通信对高效基本函数的硬件实现的需求日益增加,例如三角函数、乘法、开方以及反三角函数等复杂运算[1].CORDIC(Coordinate Rotation Digital Computer)算法是一种有效的,可以将这些复杂运算转换为简单的移位操作和加法操作的算法[2,3].

免缩放因子的 CORDIC 算法由于不需要 Z 通路,并且不需要缩放因子的补偿而倍受关注.文献[4]中,作者最先提出了无缩放因子的旋转算法,用以实现吉文斯旋转,但是其收敛范围非常小,只有$[0,\pi/8]$.随后,很多研究者针对此问题进行了研究改进.

文献[5]中提出将某些固定的角度重复迭代多次,从而扩大算法的收敛范围,然而重复迭代会增加算法的迭代次数.文献[6]中,作者使用了区间折叠技术,将整个坐标区间都映射到$[0,\pi/8]$,虽然该算法在旋转过程中可以免去动态缩放因子的计算,但是还是会产生常数缩放因子$1/\sqrt{2}$,并且随着位宽的增加,其迭代次数呈指数增加,同时面积消耗大也是其缺点之一.文献[7]中,作者用 Booth 编码技术来减少算法的面积消耗、功耗和延时,同时用两次传统的 CORDIC 迭代代替文献[6]中的区间折叠技术,将算法的收敛范围由$[0,\pi/8]$扩展到$[0,\pi/2]$.

文献[8]中作者提出了扩展因子预编码的两阶段 CORDIC 旋转算法,该算法虽然将 $i>(m-1)/2$ 之后相邻的旋转进行了两两合并,但是对于 $i<(m-1)/2$ 的部分还需要缩放因子的补偿和额外的 RAM 资源来存放缩放因子.

文献[9]中作者选择合适的泰勒级数来近似估算旋转矩阵,并且用区间折叠技术将整个圆周映射到区间$[0,\pi/4]$内,该算法完全不需要缩放因子补偿,但是和文献[8]中的算法一样,在高位宽的情况下,其迭代次数的消耗也是惊人的.

收稿日期:2012-10-14;修回日期:2014-02-07;责任编辑:孙瑶
基金项目:国家自然科学基金(No.61173037,No.61300037);广东省教育部科技部产学研合作项目(No.2011B090400356)

综上所述,多数研究在扩大免缩放因子收敛范围的同时,也增加了算法复杂度和算法迭代次数.本文针对无缩放因子 CORDIC 算法中迭代次数过多的问题对其进行研究,在不需要缩放因子补偿的前提下减少了算法的迭代次数,扩大了算法收敛范围.

2 免缩放因子 CORDIC 算法原理

与传统的 CORDIC 算法不同,免缩放因子 CORDIC 算法利用迈克劳林公式将 sin 和 cos 函数展开成级数,由文献[3]可知,当 i 的取值范围在式(1)范围内时,单步旋转操作可表示为式(2)所示形式:

$$\left[\frac{N-2.585}{3}\right] \leqslant i \leqslant N-1 \tag{1}$$

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} \cos(\alpha_i) & -\cos(\alpha_i) \\ \sin(\alpha_i) & \cos(\alpha_i) \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} 1-2^{-(2i+1)} & -2^{-i} \\ 2^{-i} & 1-2^{-(2i+1)} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{2}$$

这种迭代形式易于实现,只需要移位操作和加法/减法操作即可.而且每次迭代的向量长度不变,不需要扩展因子补偿.但缺点是收敛范围太小、迭代次数过多,导致算法延时较大.

3 区间折叠与双步旋转策略

3.1 区间折叠技术

向量旋转算法一般需要处理周期为 2π 的旋转角,常规 CORDIC 算法的收敛域约为[-99.88°,99.88°],那么需要多旋转两次 $i=0(\pi/4)$ 的角度,才能把其收敛域扩展至$[0,2\pi]$.文献[6]中将$[0,2\pi]$的区间映射到[0,$\pi/8$],但是这会产生常数缩放因子 $1/\sqrt{2}$.文献[8]将$[-\pi,\pi]$的区间映射到$[0,\pi/8]$,但这需要进行输入角度符号的判断.本章提出的免缩放因子双步旋转 CORDIC 算法将$[0,2\pi]$的区间映射到$[0,\pi/4]$内,这样既不会产生任何缩放因子,也不需要进行输入角度符号的判断.该算法将整个圆周$[0,2\pi]$划分为 8 个区间(如图 1 所示),临界角可属于相邻两区间中的任意一个.再用映射函数将所有区间的角度映射到$[0,\pi/4]$内,即区间 A 中,我们把这一过程称为区间折叠技术.

设初始旋转向量为(x_0,y_0),(x_k,y_k)表示经过旋转后的目标向量,下标 k 表示旋转角度 θ 所在的区间 ($k\in$(A,B,C,D,E,F,G,H)).那么各个区间的目标向量可以表示为:

$$\begin{pmatrix} x_A \\ y_A \end{pmatrix} = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \boldsymbol{R}\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

$$\begin{pmatrix} x_B \\ y_B \end{pmatrix} = \left(\boldsymbol{R}\begin{pmatrix} x_0 \\ -y_0 \end{pmatrix}\right)^{\mathrm{T}}$$

$$\begin{pmatrix} x_C \\ y_C \end{pmatrix} = \boldsymbol{R}\begin{pmatrix} x_0 \\ -y_0 \end{pmatrix}^{\mathrm{T}}, \begin{pmatrix} x_D \\ y_D \end{pmatrix} = \left(\boldsymbol{R}\begin{pmatrix} -x_0 \\ -y_0 \end{pmatrix}^{\mathrm{T}}\right)^{\mathrm{T}}$$

$$\begin{pmatrix} x_E \\ y_E \end{pmatrix} = \boldsymbol{R}\begin{pmatrix} -x_0 \\ -y_0 \end{pmatrix}, \begin{pmatrix} x_F \\ y_F \end{pmatrix} = \left(\boldsymbol{R}\begin{pmatrix} -x_0 \\ y_0 \end{pmatrix}\right)^{\mathrm{T}}$$

$$\begin{pmatrix} x_G \\ y_G \end{pmatrix} = \boldsymbol{R}\begin{pmatrix} -x_0 \\ y_0 \end{pmatrix}^{\mathrm{T}}, \begin{pmatrix} x_H \\ y_H \end{pmatrix} = \left(\boldsymbol{R}\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}^{\mathrm{T}}\right)^{\mathrm{T}}$$

由上式可以看出,将$[0,2\pi]$的区间映射到$[0,\pi/4]$内,没有产生任何缩放因子,也没有必要进行输入角度符号的判断.区间折叠模块的结构框图可由 2 表示.输入角度 θ 进入角度映射模块后,先判断输入角度属于哪个区间,即产生对应的 S 信号;再用映射函数将其转换到区间$[0,\pi/4]$内,得到 φ,再根据 S 信号调整输入向量得到(x',y').

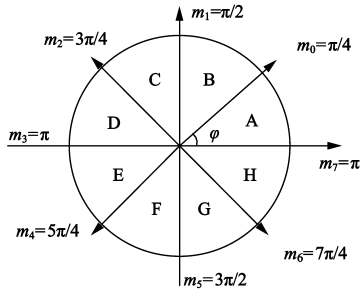

图1 区间划分

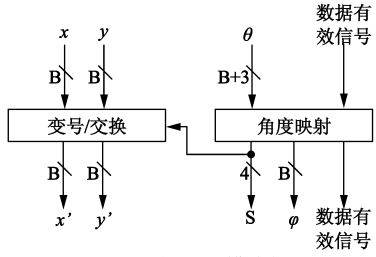

图2 区间折叠模块框图

S 信号是用来调整基本旋转 $\boldsymbol{R}(\varphi)$的输入坐标和输出坐标的控制信号,各个区间所对应的 S 值如表 1 所示.其中最低位 S1 表示在旋转前输入 x_{in}坐标值是否要取反,为 0 表示不需要取反,为 1 表示要取反;S2 表示在旋转前输入 y_{in}坐标值是否要取反,为 0 表示不需要取反,为 1 表示要取反;S3 表示在旋转前输入 x_{in}和 y_{in}坐标是否要交换,为 0 表示不需要交换,为 1 表示要交换;S4 表示在旋转后输出的 x_{out}和 y_{out}坐标是否要交换,为 0 表示不需要交换,为 1 表示要交换.因此,在进行基本旋转 $\boldsymbol{R}(\varphi)$之前根据表 1 中的 S3、S2、S1 信号对输入坐标进行调整,旋转结束后根据 S4 信号对旋转结果进行相应的调整就可得到对应区间的目标向量.

我们将区间 A 中的旋转认为是基本旋转 $\boldsymbol{R}(\varphi)$,即旋转角度为 φ、旋转矩阵为 $\boldsymbol{R}=\begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix}$的

旋转.那么由式(4)可知其他任意区间中的旋转均可以通过基本旋转 $\boldsymbol{R}(\varphi)$来完成,只需要相应的调整输入向量的坐标和旋转后输出向量的坐标即可得到各个区间的目标向量.

表 1　θ 所在区间与控制信号 S 的关系

区间	S4	S3	S2	S1
A	0	0	0	0
B	1	0	1	0
C	0	1	1	0
D	1	1	1	1
E	0	0	1	1
F	1	0	0	1
G	0	1	0	1
H	1	1	0	0

完整的 CORDIC 算法输入输出数据流如图 3 所示,其中(x_0, y_0, θ)为 CORDIC 算法实际输入,根据 θ 所在区间,区间折叠模块将输入向量经过相应的调整,得到基本旋转 $\boldsymbol{R}(\varphi)$的输入,即$(x_{\text{in}}, y_{\text{in}}, z_{\text{in}})$,该向量经过基本旋转模块 $\boldsymbol{R}(\varphi)$运算后得到输出向量$(x_{\text{out}}, y_{\text{out}}, z_{\text{out}})$,结果映射模块根据 S 信号对其做适当的调整,就得到相应区间的目标向量(x_k, y_k, z_k),下标 k 表示输入角度 θ 所在的区间($k \in$(A,B,C,D,E,F,G,H)).

图3　CORDIC算法输入输出数据流

3.2　双步旋转机制

双步旋转机制的核心就是将基本旋转 $\boldsymbol{R}(\varphi)$的两次旋转合并,在一次迭代中完成,以这种方式来达到减少迭代次数的目的.假设两次旋转的角度分别为 $\alpha_j = 2^{-j}$和$\alpha_k = 2^{-k}(j \leqslant k)$,那么双步旋转可由下式表示:

$$\begin{pmatrix} x_{i+2} \\ y_{i+2} \end{pmatrix} = \begin{pmatrix} \cos(\alpha_k) & -\sin(\alpha_k) \\ \sin(\alpha_k) & \cos(\alpha_k) \end{pmatrix} * \begin{pmatrix} \cos(\alpha_j) & -\sin(\alpha_j) \\ \sin(\alpha_j) & \cos(\alpha_j) \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{3}$$

其中双步旋转矩阵可以表示为:

$$\begin{pmatrix} \cos(\alpha_k) & -\sin(\alpha_k) \\ \sin(\alpha_k) & \cos(\alpha_k) \end{pmatrix} \begin{pmatrix} \cos(\alpha_j) & -\sin(\alpha_j) \\ \sin(\alpha_j) & \cos(\alpha_j) \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \varepsilon \end{pmatrix} \tag{4}$$

其中,$\alpha = 1 - 2^{-1}(2^{-2k} + 2^{-2j}) + 2^{-2(k+j+1)} - 2^{-(k+j)}$

$\beta = -(2^{-k} + 2^{-j} + 2^{-2(2k+j+1)} - 2^{-(k+2j+1)})$

$\gamma = 2^{-k} + 2^{-j} + 2^{-(2k+j+1)} - 2^{-(k+2j+1)}$

$\varepsilon = 1 - 2^{-1}(2^{-2k} + 2^{-2j}) + 2^{-2(k+j+1)} - 2^{-(k+j)}$

当 $j = k$ 时,是以 α_s 作为旋转角度的,这时的 j 和 k 均小于或者等于 $i_{\min}$,因此双步旋转矩阵可以简化为:

$$\begin{pmatrix} \cos(\alpha_k) & -\sin(\alpha_k) \\ \sin(\alpha_k) & \cos(\alpha_k) \end{pmatrix} \begin{pmatrix} \cos(\alpha_j) & -\sin(\alpha_j) \\ \sin(\alpha_j) & \cos(\alpha_j) \end{pmatrix} = \begin{pmatrix} X & -2^{-k+1} \\ 2^{-k+1} & Y \end{pmatrix} \tag{5}$$

其中,$X = 1 - 2^{-2k+1} + 2^{-2(2k+1)}$

$Y = 1 - 2^{-2k+1} + 2^{-2(2k+1)}$

当 $j \geqslant (B-1)/2$ 时,$k + j$、$2k$ 以及 $2j$ 均大于 B,任何 B 位宽的数向右移 $k + j$、$2k$ 或者 $2j$ 位均等于机器零.故双步旋转矩阵可简化为:

$$\begin{pmatrix} \cos(\alpha_k) & -\sin(\alpha_k) \\ \sin(\alpha_k) & \cos(\alpha_k) \end{pmatrix} \begin{pmatrix} \cos(\alpha_j) & -\sin(\alpha_j) \\ \sin(\alpha_j) & \cos(\alpha_j) \end{pmatrix} = \begin{pmatrix} 1 & -2^{-k+1} \\ 2^{-k+1} & 1 \end{pmatrix} \tag{6}$$

经过上面的推导转换,两步旋转操作可以在一次迭代中完成,并且只需要加法和移位操作,在 $j = k$ 和 $j \geqslant (B-1)/2$ 这两种情况下,算法的结构还可进一步的简化,减少了算法延时.该策略相对于传统的无缩放因子 CORDIC 算法能大大减少其迭代次数,减少延时.

4　系统结构

免缩放因子双步旋转 CORDIC 算法的硬件实现电路大致分为区间折叠、双步旋转和结果映射三大部分,算法的数据通路如图 4 所示.其中区间折叠模块主要完成将输入角度 θ 映射到区间 A,并产生相应的控制信号 S 以及计算对应的角度 φ,除此之外还要根据 S 信号相应地调整输入向量(x_i, y_i);双步旋转模块主要是完成[0,π/4]范围内角度的无缩放因子 CORDIC 算法的双步旋转过程;结果映射模块则是根据控制信号 S 的最高位 S4 调整双步旋转模块的输出向量(x'_{i+2}, y'_{i+2}),如果 S4 等于 1,则交换 x 和 y 坐标值,否则,不交换,从而得到输入角度所对应的实际旋转结果(x_{i+2}, y_{i+2}).

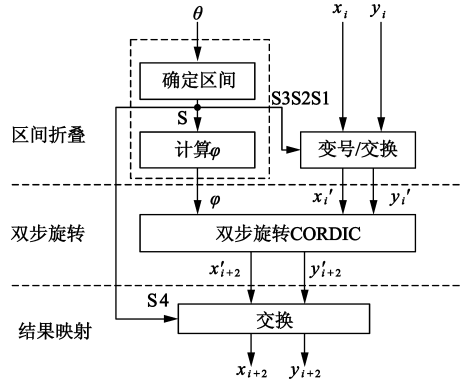

图4　双步旋转CORDIC算法的数据通路

5 性能分析

5.1 面积开销

在 CORDIC 算法的面积评估方面,经常采用全加器个数来近似估算算法的面积消耗.本文的免缩放因子双步旋转(Double-Step Scaling Free,DSSF)CORDIC 算法总共约需要 445 个全加器.而原始的 SF CORDIC 算法由于每次只处理一次旋转操作,其所需的面积相对于本文的 DSSF CORDIC 算法要少消耗约 256 个全加器.文献[6]中的 MVSF (Modified Virtually Scaling-Free)CORDIC 算法由于采用了 14 级流水设计以及需要缩放因子补偿,其需要的全加器约为 1000 个,是本算法的 2.25 倍.文献[8]中,2S-PCS 算法需要 712 个全加器,是本文的 DSSF 算法的 1.6 倍.

5.2 收敛性能

本节将[0,π/4]即[0,0.7854]范围内,步长为0.0013的所有角度作为初始 Z 输入,并以(x,y) = (1,0)做为初始输入向量,并记录下位宽分别为 N = 8,16 和 24 时的所有输入角度情况下所需要的迭代次数,并求均值,用以统计并分析 DSSF CORDIC 算法的迭代次数的分布以及数量.

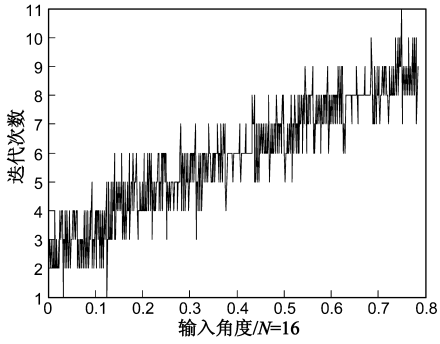

图5 N=16时的[0, π/4]范围内角度的迭代次数分布图

图 5 是 N = 16 时的迭代次数分布图,由图可以看出随着旋转角度的增加,所需的迭代次数也呈线性增加的趋势.这是由于在位宽大小一定的条件下,每次旋转的最大常数角 $2^{-i_{\min}}$是固定的,那么旋转角度的增加就意味着需要重复更多次的 $2^{-i_{\min}}$迭代,从而会导致迭代次数增加.表 2 是原始的 SF CORDIC 算法、MVSF CORDIC 算法[6]以及本文提出的 DSSF CORDIC 算法在不同位宽大小条件下的平均迭代次数.

实验证明,相对于 SF CORDIC 算法,本章提出的 DSSF CORDIC 算法的迭代次数减少了至少 50%.但是在位宽较大时(如 $N \geqslant 24$)其所需的迭代次数依然较多.但仍比 MSVF CORDIC 算法少大约 3 次迭代.和文献[8]相比,本文 DSSF 算法需要消耗较多的迭代次数,这是由于文献[8]用常规的 CORDC 算法做旋转,直到旋转角度在 SF CORDIC 算法的收敛范围之内.这样,其就以面积消耗为代价减少了迭代次数.

表 2 迭代次数比较

算法	N = 8	N = 16	N = 24
SF	3.16	11.26	53.78
MVSF[6]	2.16	7.63	32.13
DSSF	1.84	5.88	29.13

5.3 精度分析

图 6 是 N = 16(1 位整数位,1 位符号位,14 位小数位)时 x 和 y 轴的误差分布图.由图可以看出最坏情况下的计算精度能达到小数点后第 8 位,而平均计算精度能达到小数点后约 13 位;图 7 是 N = 24(1 位整数位,1 位符号位,22 位小数位)时 x 和 y 轴的误差分布图,此时最坏情况下的计算精度能达到小数点后第 14 位,平均计算精度也能达到小数点后约 21 位.

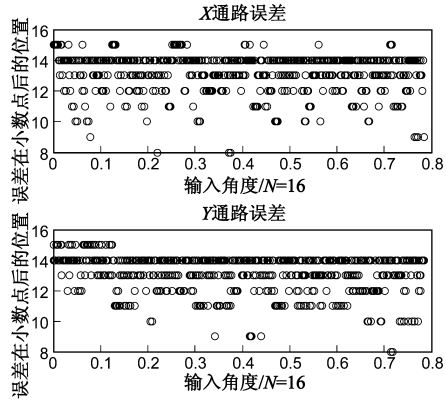

图6 DSSF算法N=16时的误差分布图

常规 CORDIC 算法需使用($n + \log_2 n + 2$)位数据通路[10]才能达到 n 位精度,如果用 16 位数据通路,只能达到约 11 位精度,去掉小数点前两位,平均计算精度只能达到小数点后 9 位,比本文的 DSSF CORDIC 算法低 4 位;如果用 24 位数据通路,计算精度只能达到小数后约 18 位,比本文的 DSSF CORDIC 算法低 3 位.X、Y 通路的误差主要由每次迭代过程中的圆整操作引起,本文的 DSSF CORDIC 算法精度高的优势主要得益于其将两步旋转操作合并在一次迭代中完成,并且去除了 Z 通路和缩放因子补偿操作,减少了每次迭代中的圆整操作,从而提高了计算精度.

6 总结

本文针对 SF CORDIC 算法迭代次数过多、收敛范围有限以及计算精度有限等问题,通过分析 SF CORDIC 算法旋转角度的特点,结合圆周区间的对称性,提出了免缩放因子双步旋转 CORDIC 算法.该算法使用区间折叠技术,将整个圆周区间都映射到[0,π/4]内,该范围内

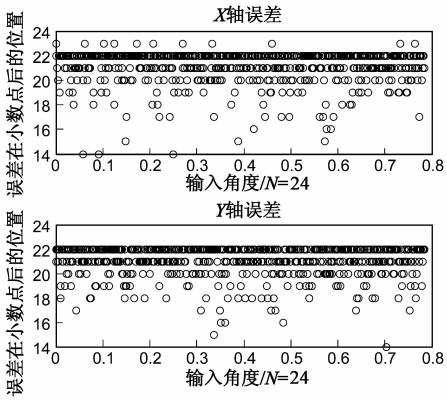

图7　DSSF算法N=24时的误差分布图

的旋转结果经过一定规则的调整,就能得到实际的旋转向量;使用双步旋转策略,在每次迭代中处理两步旋转操作,大大减少了免缩放因子 CORDIC 算法的迭代次数;并且合并两次相邻迭代次数减少了圆整操作,从而提高了算法的计算精度.实验结果表明本文提出的免缩放因子双步旋转 CORDIC 算法达到了减少迭代次数的目的,并在计算精度、迭代次数以及面积消耗等方面均优于文献[6]中的 MVSF 算法.

参考文献

[1] 雷元武,窦勇,倪时策,周杰.定制 VLIW 结构实现四精度浮点基本函数[J].电子学报,2012,40(9):1715 – 1722.
LEI Yuan-wu, DOU Yong, NI Shi-ce, ZHOU Jie. A special purpose VLIW structure to implement quadruple precision floating point elementary function [J]. Acta Electronica Sinica, 2012,40(9):1715 – 1722. (in Chinese)

[2] 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)

[3] K Maharatna, A Troya, S Banerjee, E Grass. Virtually scaling-free adaptive CORDIC rotator [J]. IEE Proceedings Computers & Digital Techniques, 2004, 151(6): 448 – 456.

[4] Anindya S Dhar, Swapna Banerjee. An array architecture for fast computation of discrete hartley transform [J]. IEEE Transactions on Circuits and Systems, 1991, 38(9): 1095 – 1098.

[5] X Hu, R G Harber, S C Bass. Expanding the range of convergence of the CORDIC algorithm [J]. IEEE Transaction on Computers, 1991, 40(1): 13 – 21.

[6] K Maharatna, S Banerjee, E Grass, M Krstic, A Troya. Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture [J]. IEEE Transaction on Circuits Systems for Video Technology, 2005, 15(11): 1463 – 1474.

[7] Francisco J Jaime, Miguel A Sánchez, Javier Hormigo, et al. Enhanced scaling-free CORDIC [J]. IEEE Transactions on Circuits and Systems—I: Regular Papers, 2010, 57(7): 1654 – 1662.

[8] 牟胜梅,杨晓东.扩展因子与编码的两阶段 CORDIC 旋转算法 2S-PCS [J].计算机学报,2011,31(4):729 – 737.
MOU Sheng-mei, YANG Xiao-dong. A two-step CORDIC rotation algorithm with pre-coded scale factors [J]. Chinese Journal of Computers, 2011, 31(4): 729 – 737. (in Chinese)

[9] Supriya Aggarwal, Pramod K Meher, Kavita Khare. Area-time efficient scaling-free CORDIC using generalized micro-rotation selection [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2012, 20(8): 1542 – 1546.

[10] Kotak, Cavallaro J R. Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special purpose processors [J]. IEEE Transactions on Computers, 1993, 42(7): 769 – 777.

作者简介

徐　成　男,1962 年 12 月出生,湖北蕲春县人,教授、博士生导师.1983 年、1986 年和 2006 年分别获得理学学士、工学硕士和工学博士学位.现主要研究方向为嵌入式系统、数字视频处理和自动测试与控制.
E-mail:cheng_xu@yeah.net

秦云川　男,1983 年 4 月出生,重庆合川人,博士研究生.分别于 2005 和 2008 年在湖南大学获得工学学士和工学硕士学位.现主要研究方向为嵌入式系统的体系结构和信息安全.
E-mail:qinyunchuan@hnu.edu.cn