# 扩展因子预编码的两阶段 CORDIC 旋转算法 2S-PCS

牟胜梅[1) ] 杨晓东[2)]

[1)](海军工程大学计算机工程系 武汉 430033)

[2)](国防科技大学计算机学院 长沙 410073)

**摘 要** CORDIC 算法常用于高效地用硬件实现向量旋转操作,如何减少迭代次数并保持扩展因子计算与补偿的简单性是算法的难点.文中提出一种表驱动的 2S-PCS 算法,其流水线较短且扩展因子可预先计算并优化编码.算法首先将[$-\pi,\pi$]内的旋转角映射到[$0,\pi/4$]内,并产生初值调整和旋转方向控制信号.之后的旋转过程分为两阶段,步 1 进行扩展因子可变的大角度旋转,使剩余旋转角进入步 2 的收敛域.步 1 的迭代系数和扩展因子均由映射后旋转角的高字段作为地址查表获得.步 2 扩展因子恒为 1,迭代系数直接由旋转角的二进制编码决定.整个过程不需 $z$ 通道和扩展因子计算通道,节省了面积开销.2S-PCS 利用角度分解算法生成步 1 的迭代系数,约束非零系数的位置,并对扩展因子进行基 4-Booth 编码,同时合并相邻的计算量小的迭代,以减少流水线级数.2S-PCS 算法利用 CSA 实现三数累加,同时忽略超出精度表示范围的表达式,以减少延迟、降低复杂性.短流水线还可减少计算通路的圆整(rounding)误差,提高精度.2S-PCS 算法克服了全字段查表可扩展性差的问题,入口数随数据精度的增加而缓慢增长,可扩展性好.当采用 28 位数据通路时,与常规 CORDIC 算法相比,2S-PCS 算法的流水线级数减少约 38%,面积减少约 27.9%,精度提高 3 位左右,具有明显的性能优势.

**关键词** 向量旋转;CORDIC 算法;查找表;免扩展;角度重编码

**中图法分类号** TP302 **DOI 号**: 10.3724/SP.J.1016.2011.00729

# A Two-Step CORDIC Rotation Algorithm with Pre-Coded Scale Factors

MOU Sheng-Mei[1)] YANG Xiao-Dong[2)]

[1)](*Department of Computer Engineering, Naval University of Engineering, Wuhan* 430033)

[2)](*School of Computer Science, National University of Defense Technology, Changsha* 410073)

**Abstract** CORDIC algorithm has been widely used for efficient implementation of vector rotation operations in hardware. However, it's hard to achieve small iteration numbers while not compromising the ease of scale factor computing and compensation. This paper proposes a novel table-driven CORDIC rotation algorithm with predictable scale factors and short pipelines. It maps the input angle during [$-\pi,\pi$] into [$0,\pi/4$] and generates signals to modify the initial iteration value and control the rotation direction. The rotation process is composed of two steps. The coefficients and scale factors of step1 are acquired through lookup table indexed by the MSB (Most Significant Bits) of the mapped angle. After large-angle rotations of step1, the residual angle is brought into the convergence region of step2. The iteration of step2 is scaling-free, with the coefficients directly gotten from the binary representation of the residual angle. During such two processes, no further arithmetic computation in the angle approximation ($z$ datapath) and scale factor computing datapath is required. The coefficients of step1 are generated by our angle de-

收稿日期:2008-07-13;最终修改稿收到日期:2010-09-07.本课题得到海军工程大学自然科学基金(HGDQNJJ10028)资助.**牟胜梅**,女,1979 年生,博士,讲师,主要研究方向为计算机系统结构、嵌入式系统设计和算法硬件加速.E-mail: hpcs_msm@163.com. **杨晓东**,男,1936 年生,教授,博士生导师,主要研究领域为计算机系统结构、分布与并行处理及 RAS.

composition algorithm, with two coefficients in one group, at most one non-zero value each group, and the scale factors are pre-computed and coded with radix-4 Booth recoding algorithm. In addition, some adjacent simple micro rotations are combined into one. All these techniques effectively reduce the iteration number, which is also beneficial to high accuracy due to less rounding errors. Moreover, the algorithm uses carry-save adders to accumulate three operands and omits expressions of machine zero, which reduces the latency and complexity of the system greatly. The MSB indexed lookup table gets over the poor scalability of full-word indexed lookup table, and entries grow slowly with accuracy. When using 28-bit datapath, the algorithm requires 38% less pipeline stages and 27.9% less area consumption compared to those of the conventional CORDIC, while achieving 3 more bits accuracy, which shows great performance superiority.

**Keywords** vector rotation; CORDIC algorithm; lookup table; scaling free; angle recoding

## 1 引 言

向量旋转是数字信号处理和科学计算等领域的常用操作,CORDIC(COordinate Rotational DIgital Computer)算法[1]因计算简单、结构规整而被用于硬件实现向量旋转.向量旋转可分为两类,一类是旋转角取值随机性强、选取范围较广且不可预知的情况,本文称其为I类应用;另一类是旋转角数目有限且可预知的旋转操作,称为II类应用.II类应用算法优化空间较大,可利用搜索算法[2-5]预先计算迭代系数,求得扩展因子并对其进行优化编码,以较少的迭代次数达到精度要求.但对于I类应用,较少的迭代次数与简单的扩展因子计算和补偿操作相矛盾,算法较难优化.

常规CORDIC算法的旋转因子只与精度 $n$ 有关,易计算与补偿,但迭代次数较多,计算延迟较大.为减少迭代次数,一些优化策略被采用,如合并两次基本旋转[6],利用冗余算术、采用CSA(Carry-Save Adder)减少加法延迟[7-8];允许迭代系数为0,旁路不必要的迭代;采用高基CORDIC算法[7]等.但这些算法多以增加扩展因子的计算复杂性为代价.例如:允许迭代系数取0,使得扩展因子不固定,必须伴随旋转操作更新扩展因子,增加了计算开销.目前面向I类应用的改进型向量旋转CORDIC算法大多需动态计算扩展因子,MVSF算法[9]虽支持固定旋转因子($\pm1$ 或 $\pm1/\sqrt{2}$),但仅适用于中低精度(位宽 $n\leq17$).随着 $n$ 的增加,迭代次数呈指数规律剧增.文献[13]采用分段符号预测加校正的方式提高了系统的并行性,但计算模式不规整.

本文提出的2S-PCS算法面向I类应用,兼顾了迭代次数与扩展因子的计算复杂性.旋转过程分为两个阶段,通过步1的有限几次大角度旋转,使剩余旋转角进入步2的收敛域.步1利用旋转角的高位进行查表,获得迭代系数(旋转方向)和扩展因子.其迭代表达式与常规CORDIC算法相同,但迭代系数可取{0,−1,1},以旁路不必要的旋转.步2采用扩展因子恒为1的迭代算法,且直接将旋转角的二进制编码作为迭代系数.整个过程不需扩展因子与迭代系数计算通路.扩展因子采用基4-Booth编码,每两位组成的位片至多含一个非零位,以减少扩展因子补偿操作的流水线级数.这些方法有效地减少了流水线级数、计算延迟和电路的面积开销.

本文第2节介绍常规CORDIC算法;第3节介绍SF CORDIC算法与Step2的实现;第4节阐述Step1的实现思路,包括旋转角区间压缩、角度分解以及查找表的生成;第5节描述算法实现电路的系统结构;第6节对算法性能进行模拟与分析;最后一节总结全文.

## 2 常规CORDIC算法

CORDIC是一种通用迭代算法,可在线性、圆坐标和双曲坐标模式下进行定向与旋转操作,用于各类初等函数求值.其中向量旋转操作应用最为广泛.CORDIC算法将旋转角分解为若干基本旋转角 $\alpha^{(i)}$,每次旋转仅由移位和加法操作完成.设向量 $\boldsymbol{E}^{(i)}(x^{(i)},y^{(i)})$ 逆时针旋转角度 $\alpha^{(i)}$ 得到向量 $\boldsymbol{E}^{(i+1)}(x^{(i+1)},y^{(i+1)})$,则两向量的端点坐标间存在如下对应关系

$$\begin{pmatrix}x^{(i+1)}\\y^{(i+1)}\end{pmatrix}=\begin{pmatrix}\cos\alpha^{(i)} & -\sin\alpha^{(i)}\\ \sin\alpha^{(i)} & \cos\alpha^{(i)}\end{pmatrix}\begin{pmatrix}x^{(i)}\\y^{(i)}\end{pmatrix}$$

$$=\cos\alpha^{(i)}\begin{pmatrix}1 & -\tan\alpha^{(i)}\\ \tan\alpha^{(i)} & 1\end{pmatrix}\begin{pmatrix}x^{(i)}\\y^{(i)}\end{pmatrix}$$

用变量 $z$ 记录向量当前位置到目标位置的转角，$z^{(i+1)}=z^{(i)}-\alpha^{(i)}$（逆时针旋转为正），$z^{(0)}$ 表示初始向量到目标位置的转角. 经过若干次迭代，当 $z^{(i+1)}\to 0$ 时，向量便旋转到目标位置.

多数 CORDIC 算法在迭代过程中忽略 $\cos\alpha^{(i)}$ 因子，令 $\tan\alpha^{(i)}=d_i 2^{-i}$（$d_i\in\{-1,1\}$），利用式(1)简化迭代，故迭代结束时需对终值 $(x^{(n)}\ \ y^{(n)})^{\mathrm{T}}$ 进行校正，$(x^f\ \ y^f)^{\mathrm{T}}=K(x^{(n)}\ \ y^{(n)})^{\mathrm{T}}$. 扩展因子 $K=\prod_i\cos\alpha^{(i)}=1/\prod_i\sqrt{1+d_i^2\cdot 2^{-2i}}=1/\prod_i\sqrt{1+2^{-2i}}$. 由于扩展因子只取决于迭代次数 $n$（数据表示精度），因此不需在旋转过程中临时计算. 常规 CORDIC 算法的收敛域约为 $[-99.88°, 99.88°]$.

$$\begin{aligned}\begin{pmatrix}x^{(i+1)}\\ y^{(i+1)}\end{pmatrix}&=\begin{pmatrix}1 & -\tan\alpha^{(i)}\\ \tan\alpha^{(i)} & 1\end{pmatrix}\begin{pmatrix}x^{(i)}\\ y^{(i)}\end{pmatrix}\\&=\begin{pmatrix}1 & -d_i\cdot 2^{-i}\\ d_i\cdot 2^{-i} & 1\end{pmatrix}\begin{pmatrix}x^{(i)}\\ y^{(i)}\end{pmatrix}\end{aligned}\tag{1}$$

$$z^{(i+1)}=z^{(i)}-\alpha^{(i)}=z^{(i)}-d_i\tan^{-1}2^{-i},\quad 0\le i\le n-1$$

$$d_i=\mathrm{sign}(z^{(i)})=\begin{cases}1, & z^{(i)}\ge 0\\ -1, & z^{(i)}<0\end{cases}$$

## 3　SF 算法(Scaling-free CORDIC)[10]

与常规 CORDIC 算法不同，SF 算法取 $\alpha^{(i)}=2^{-i}$，直接将 $\cos\alpha^{(i)}$ 和 $\sin\alpha^{(i)}$ 表示为易于迭代的形式. 因此，每次迭代过程与真正的旋转操作等价，向量长度不变，不需进行扩展因子补偿.

假设变量 $x,y$ 均用 $m$ 位补码表示，角度 $z$ 含 $n$ 位小数，则受操作数表示精度限制，当 $\alpha^{(i)}-\sin\alpha^{(i)}<2^{-n}$ 时，$\sin\alpha^{(i)}\approx\alpha^{(i)}=2^{-i}$，$\sin(\alpha^{(i)}/2)\approx\alpha^{(i)}/2=2^{-(i+1)}$. 故

$$\begin{aligned}\cos(d_i\alpha^{(i)})&=1-2\sin^2(d_i\alpha^{(i)}/2)\\&\approx 1-d_i^2\times 2\times 2^{-2(i+1)}\\&=1-|\,d_i\,|\,2^{-(2i+1)}.\end{aligned}$$

利用麦克劳林(Maclaurin)公式[11]将 $\sin x$ 展成级数，利用误差项表达式可求得满足 $\alpha^{(i)}-\sin\alpha^{(i)}<2^{-n}$ 的 $i$ 值下界(非下确界)：$\lceil(n-\log_2^6)/3\rceil$. 实际应用中可验证是否可取 $\lfloor(n-\log_2^6)/3\rfloor$. 为简化描述，记 $i$ 的最小值为 $t$. 故算法迭代范围为 $t\le i\le n-1$. 图 1 显示了 $n$ 与 $t$ 的对应关系.

本文步 2 采用单向旋转，$d_i$ 的符号由步 1 的余角 $\phi_r$ 的符号位 $s_r$ 与控制信号 $s_1$（取决于旋转角所在区间，详见第 4 节）共同决定，初始旋转角取 $z^{(0)}=|\phi_r|$，其二进制编码决定相应的 $|d_i|$，$z$ 通路可省.

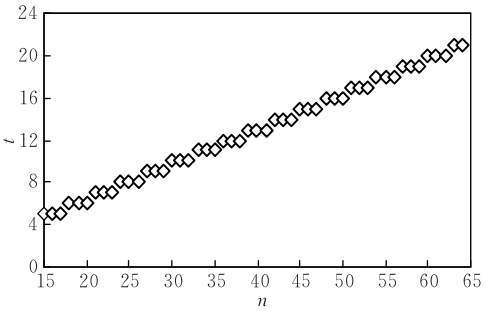


图 1　$n$ 与 $t$ 的对应关系

算法迭代公式表示如下

$$\begin{pmatrix}x^{(i+1)}\\ y^{(i+1)}\end{pmatrix}=\begin{pmatrix}1-|\,d_i\,|\,2^{-(2i+1)} & -d_i\cdot 2^{-i}\\ d_i\cdot 2^{-i} & 1-|\,d_i\,|\,2^{-(2i+1)}\end{pmatrix}\begin{pmatrix}x^{(i)}\\ y^{(i)}\end{pmatrix}$$

$$z^{(i+1)}=z^{(i)}-|\,d_i\,|\cdot 2^{-i},\ d_i\in\{0,S\},$$

$$S=\begin{cases}1, & s_1\oplus s_r=0\\ -1, & \text{否则}\end{cases}.$$

当 $2i+1\ge m$，即 $i\ge(m-1)/2$ 时，$2^{-(2i+1)}$ 为机器 0，上式可简化为与常规 CORDIC 算法相同的迭代表达式：

$$\begin{pmatrix}x^{(i+1)}\\ y^{(i+1)}\end{pmatrix}=\begin{pmatrix}1 & -d_i\cdot 2^{-i}\\ d_i\cdot 2^{-i} & 1\end{pmatrix}\begin{pmatrix}x^{(i)}\\ y^{(i)}\end{pmatrix}.$$

SF 算法的收敛域较小，仅为 $[-f(n),f(n)]$，$f(n)=\sum_{i=t}^{n-1}2^{-i}$，故需先通过若干次迭代将旋转角调整至 SF 算法收敛域内. MVSF 算法[9]通过重复执行多次 $i=t$ 的 SF 迭代实现该过程，但随 $n$ 的增加，迭代次数呈指数增加. 2S-PCS 算法则在步 1 借助查找表实现扩展因子可变的旋转操作，迅速将旋转角调整至 SF 算法收敛域内.

## 4　迭代系数与扩展因子预编码

### 4.1　旋转角区间压缩(range reduction)

向量旋转算法一般需处理周期为 $2\pi$ 的旋转角. 若使用常规 CORDIC 算法，需增加两次 $i=0(\pi/4)$ 的旋转，才能将其收敛域扩展至 $[-\pi,\pi]$. 2S-PCS 算法为减少第一阶段的查找表入口数，首先通过区间压缩将初始旋转角 $\theta\in[-\pi,\pi]$ 映射到 $[0,\pi/4)$ 内. 图 2 将旋转角 $\theta\in[-\pi,\pi]$ 划分为 $A\left(\left[0,\frac{\pi}{4}\right]\right)$、$B\left(\left[\frac{\pi}{4},\frac{\pi}{2}\right]\right)$、$C\left(\left[\frac{\pi}{2},\frac{3\pi}{4}\right]\right)$、$D\left(\left[\frac{3\pi}{4},\pi\right]\right)$、$E\left(\left[-\pi,-\frac{3\pi}{4}\right]\right)$、$F\left(\left[-\frac{3\pi}{4},-\frac{\pi}{2}\right]\right)$、$G\left(\left[-\frac{\pi}{2},-\frac{\pi}{4}\right]\right)$、$H\left(\left[-\frac{\pi}{4},0\right]\right)$ 8 个区间（临界角可属于两区间任

一),利用映射函数 $f$ 将 $\theta$ 映射至 $\phi\in[0,\pi/4]$.

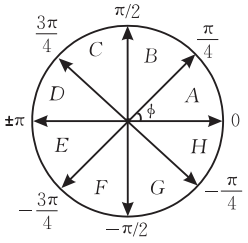


图 2 区间划分与映射

$$\phi=f(\theta)=\begin{cases}\theta, & \theta\in A\\ \pi/2-\theta, & \theta\in B\\ \theta-\pi/2, & \theta\in C\\ \pi-\theta, & \theta\in D\\ \pi+\theta, & \theta\in E\\ -\theta-\pi/2, & \theta\in F\\ \pi/2+\theta, & \theta\in G\\ -\theta, & \theta\in H\end{cases}$$

记初始向量为$(x^{(0)},y^{(0)})$,旋转角 $\theta$ 位于区间 $i(i\in\{A,B,C,D,E,F,G,H\})$时的目标向量为 $(x_i,y_i)$,则

$$\begin{pmatrix}x_A\\y_A\end{pmatrix}=\begin{pmatrix}\cos\phi & -\sin\phi\\ \sin\phi & \cos\phi\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}\triangleq R_\phi\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix},$$

$$\begin{pmatrix}x_H\\y_H\end{pmatrix}=\begin{pmatrix}\cos(-\phi) & -\sin(-\phi)\\ \sin(-\phi) & \cos(-\phi)\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}$$

$$=\begin{pmatrix}\cos\phi & \sin\phi\\ -\sin\phi & \cos\phi\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}\triangleq R_{-\phi}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix},$$

$$\begin{pmatrix}x_B\\y_B\end{pmatrix}=\begin{pmatrix}\cos\left(\frac{\pi}{2}-\phi\right) & -\sin\left(\frac{\pi}{2}-\phi\right)\\ \sin\left(\frac{\pi}{2}-\phi\right) & \cos\left(\frac{\pi}{2}-\phi\right)\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}$$

$$=\begin{pmatrix}\sin\phi & -\cos\phi\\ \cos\phi & \sin\phi\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}=R_{-\phi}\begin{pmatrix}-y^{(0)}\\x^{(0)}\end{pmatrix},$$

$$\begin{pmatrix}x_C\\y_C\end{pmatrix}=\begin{pmatrix}\cos\left(\frac{\pi}{2}+\phi\right) & -\sin\left(\frac{\pi}{2}+\phi\right)\\ \sin\left(\frac{\pi}{2}+\phi\right) & \cos\left(\frac{\pi}{2}+\phi\right)\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}$$

$$=\begin{pmatrix}-\sin\phi & -\cos\phi\\ \cos\phi & -\sin\phi\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}=R_\phi\begin{pmatrix}-y^{(0)}\\x^{(0)}\end{pmatrix},$$

$$\begin{pmatrix}x_D\\y_D\end{pmatrix}=\begin{pmatrix}\cos(\pi-\phi) & -\sin(\pi-\phi)\\ \sin(\pi-\phi) & \cos(\pi-\phi)\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}$$

$$=\begin{pmatrix}-\cos\phi & -\sin\phi\\ \sin\phi & -\cos\phi\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}=R_{-\phi}\begin{pmatrix}-x^{(0)}\\-y^{(0)}\end{pmatrix},$$

$$\begin{pmatrix}x_E\\y_E\end{pmatrix}=\begin{pmatrix}\cos(-\pi+\phi) & -\sin(-\pi+\phi)\\ \sin(-\pi+\phi) & \cos(-\pi+\phi)\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}$$

$$=\begin{pmatrix}-\cos\phi & \sin\phi\\ -\sin\phi & -\cos\phi\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}=R_\phi\begin{pmatrix}-x^{(0)}\\-y^{(0)}\end{pmatrix},$$

$$\begin{pmatrix}x_F\\y_F\end{pmatrix}=\begin{pmatrix}\cos\left(-\frac{\pi}{2}-\phi\right) & -\sin\left(-\frac{\pi}{2}-\phi\right)\\ \sin\left(-\frac{\pi}{2}-\phi\right) & \cos\left(-\frac{\pi}{2}-\phi\right)\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}$$

$$=\begin{pmatrix}-\sin\phi & \cos\phi\\ -\cos\phi & -\sin\phi\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}=R_{-\phi}\begin{pmatrix}y^{(0)}\\-x^{(0)}\end{pmatrix},$$

$$\begin{pmatrix}x_G\\y_G\end{pmatrix}=\begin{pmatrix}\cos\left(\phi-\frac{\pi}{2}\right) & -\sin\left(\phi-\frac{\pi}{2}\right)\\ \sin\left(\phi-\frac{\pi}{2}\right) & \cos\left(\phi-\frac{\pi}{2}\right)\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}$$

$$=\begin{pmatrix}\sin\phi & \cos\phi\\ -\cos\phi & \sin\phi\end{pmatrix}\begin{pmatrix}x^{(0)}\\y^{(0)}\end{pmatrix}=R_\phi\begin{pmatrix}y^{(0)}\\-x^{(0)}\end{pmatrix}.$$

上述各式表明,旋转角 $\theta$ 位于区间 $A$、$C$、$E$、$G$ 时,实际旋转角取 $\phi$,位于区间 $B$、$D$、$F$、$H$ 时,实际旋转角取 $-\phi$,$x$,$y$ 的初始值作相应的交换与符号变换.因此,要计算旋转 $\theta$ 后的目标向量,需先判断 $\theta$ 所在区间,产生控制信号 $s_1s_2s_3s_4$ 调整 $x$,$y$ 的初值,并计算映射后的 $\phi$ 值.$s_1=1$ 则旋转角取 $-\phi$,即查表所得迭代系数取相反数;$s_2=1$ 则 $x$ 取相反数,$s_3=1$ 则 $y$ 取相反数,$s_4=1$ 则交换 $x$,$y$ 值.$\theta$ 所在区间与控制信号的对应关系如表 1 所示.$s_2s_3s_4$ 既可用于调整 $x$,$y$ 的初始值,也可待步 1 与步 2 全部结束且进行扩展因子补偿后再调整 $x$,$y$ 的终值,二者等价,但将 $s_2s_3s_4$ 的控制作用前移,可省掉寄存 $s_2s_3s_4$ 信号的开销.

**表 1 $\theta$ 所在区间与控制信号的对应关系**

| 区间 | $s_1$ | $s_2$ | $s_3$ | $s_4$ |
|---|---|---|---|---|
| $A$ | 0 | 0 | 0 | 0 |
| $B$ | 1 | 0 | 1 | 1 |
| $C$ | 0 | 0 | 1 | 1 |
| $D$ | 1 | 1 | 1 | 0 |
| $E$ | 0 | 1 | 1 | 0 |
| $F$ | 1 | 1 | 0 | 1 |
| $G$ | 0 | 1 | 0 | 1 |
| $H$ | 1 | 0 | 0 | 0 |

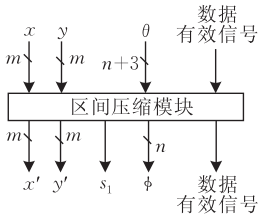


图 3 区间压缩模块

初始旋转角位于$[-\pi,\pi]$时,区间压缩过程需 3 次减法操作(减数为常数)和若干次符号判断,比初始旋转角位于$[0,2\pi]$时少用一次减法操作.区间压缩模块的结构如图 3 所示.

### 4.2　角度分解算法 CHAR

旋转角映射到 $\phi\in[0,\pi/4]$ 后，需通过若干次迭代将 $\phi$ 调整至 SF 算法的收敛域内. 2S-PCS 算法预先通过角度分解求得步 1 的迭代系数及剩余角，保存在查找表中. 角度分解属搜索带约束的最优组合问题，贪婪算法[2]可获得局部最优组合，求得的系数序列 $\{d_i\}$ 满足 $\sum_{i=0}^{n-1}|d_i|\le n/2$，但非零位置不固定，流水实现时不能有效地减少流水线级数. 本文提出一种带约束的高位角度分解算法 CHAR(Constrained High-bits Angle Recoding Algorithm)，将迭代系数两两分组，每组至多含一个非零值，流水实现时每级处理两个系数，至多进行一次迭代.

问题描述如下：给定用 $t$ 位无符号定点表示的旋转角 $\phi_H\in[0,\pi/4)$ 和无穷精度的基本旋转角 $\{\alpha(i)=a\tan 2^{-i},i=0,\cdots,t-1\}$，$t\le n$，寻找迭代系数序列 $\{d_i,i=0,\cdots,t-1\}$，$d_i\in\{0,\pm1\}$，使得：(1) $\phi_H=\sum_{i=0}^{t-1}d_i\alpha(i)+\varepsilon$，$|\varepsilon|<\alpha(t)$；(2) 迭代系数按位置顺序两两分组，每组至多含一个非零值(若 $t$ 为奇数，则最后一个系数单独成组)；(3) $|\varepsilon|$ 最小.

现给出一满足上述条件(1)、(2)的迭代系数构造算法，但求得的迭代系数未必最优，即 $|\varepsilon|$ 未必最小. $t$ 较小时可通过穷举搜索确定最优解，$t$ 较大时利用 CHAR 算法求解.

**CHAR 算法.**

初始值：$\phi(0)=\phi_H$，$\{d_i=0,i=0,\cdots,t-1\}$；

For ($k=0$; $k\le\left\lceil\frac{t}{2}\right\rceil-1$; $k{+}{+}$) {

  If $|\phi(k)|<\alpha(t)$ break;

  $|\phi(k+1)_2|=||\phi(k)|-\alpha(2k)|$;

  $|\phi(k+1)_1|=||\phi(k)|-\alpha(2k+1)|$;

  $|\phi(k+1)|=\min(|\phi(k+1)_2|,|\phi(k+1)_1|,|\phi(k)|)$;

  If $|\phi(k+1)|=|\phi(k+1)_2|$ {

    $d_{2k}=\mathrm{sign}(\phi(k))$;

    $\phi(k+1)=\phi(k)-d_{2k}\alpha(2k)$; }

  Else if $|\phi(k+1)|=|\phi(k+1)_1|$ {

    $d_{2k+1}=\mathrm{sign}(\phi(k))$;

    $\phi(k+1)=\phi(k)-d_{2k+1}\alpha(2k+1)$;}

  else $\phi(k+1)=\phi(k)$;

  }

If ($Odd(t)$ && ($|\phi(k)|\ge\alpha(t)$))

  If $\left|\left|\phi\left(\frac{t-1}{2}\right)\right|-\alpha(t-1)\right|<\left|\phi\left(\frac{t-1}{2}\right)\right|$ {

    $d_{t-1}=\mathrm{sign}\left(\phi\left(\frac{t-1}{2}\right)\right)$;

    $\phi\left(\frac{t+1}{2}\right)=\phi\left(\frac{t-1}{2}\right)-d_{t-1}\alpha(t-1)$;}

  Else $\phi\left(\frac{t+1}{2}\right)=\phi\left(\frac{t-1}{2}\right)$;

**表 2　基 4-Booth 编码前后位片值对应关系**

| $A_{2i+1}$ | $A_{2i}$ | $A_{2i-1}$ | $A'_{2i+1}$ | $A'_{2i}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | −1 | 0 |
| 1 | 0 | 1 | 0 | −1 |
| 1 | 1 | 0 | 0 | −1 |
| 1 | 1 | 1 | 0 | 0 |

**表 3　存储量与表示精度关系示例**

| $m=n$ | $t$ | 存储量/bit |
|---|---|---|
| 16 | 5 | 1404 |
| 20 | 6 | 3417 |
| 24 | 8 | 16362 |
| 28 | 9 | 37882 |
| 32 | 10 | 86135 |
| 36 | 12 | 389257 |
| 40 | 13 | 862156 |

对于 $n$ 位无符号表示的旋转角 $\phi\in[0,\pi/4]$，其高 $t$ 位便是 $\phi_H$，记 $\phi_L=\phi-\phi_H$，则 $\phi_L<2^{-t}$. 又因 CHAR 算法满足 $|\varepsilon|<\alpha(t)<2^{-t}$，故 $|\varepsilon+\phi_L|\le|\varepsilon|+|\phi_L|<2^{-(t-1)}$，即 $|\varepsilon_r|=\left|\phi_H-\sum_{i=0}^{t-1}d_i\alpha(i)+\phi_L\right|<2^{-(t-1)}$，已进入 SF 算法的收敛域. 由于搜索只需针对 $t$ 位表示的 $\phi_H$ 进行，每个角度需 $3^t$ 次穷举搜索，故高 $t$ 位全局最优搜索算法的总开销为 $(\lfloor 2^t\times\pi/4\rfloor+1)\times 3^t$，而 $n$ 位全局最优搜索算法的总开销为 $(\lfloor 2^n\times\pi/4\rfloor+1)\times 3^t$，前者仅为后者的 $2^{-(n-t)}$.

利用搜索得到的系数组合可计算扩展因子 $K=1\Big/\prod_{i=0}^{t-1}(1+d_i^2 2^{-2i})^{1/2}$，并对其进行基 4-Booth 编码，转化为 SD(Signed Digit)格式，每 2 位组成的位片至多只含一个非零值. 表 2 列出了编码前后位片值的对应关系.

### 4.3　查找表

步 1 的查找表包含 $\left\lfloor\frac{\pi}{4}\cdot 2^t\right\rfloor+1$ 个表项入口(记为 $T$)，以映射后旋转角 $\phi$ 的高 $t$ 位 $\phi_H$ 作为查表地址，每个表项保存 $t$ 个 SD 编码的迭代系数、$m$ 位 SD 编码的扩展因子(由基 4-Booth 编码转化而得)和 $n-t+1$ 位补码表示的 $\varepsilon=\phi_H-\sum_{i=0}^{t-1}d_i\alpha(i)$. 每个 SD 数需用两位表示，可用两个位串组合表示迭代系数和扩展因子，每次迭代时提取两位串中的相应位进

行译码即可. 因此查找表的总存储开销为 $M=T(2m+n+t+1)$ 位. 步 1 的剩余旋转角 $\phi_r=\phi_L+\varepsilon$. 表 3 列举了数据表示精度与存储量开销的对应关系.

## 5 系统结构与流水线划分

2S-PCS 算法实现电路的流水线级数取决于 $n$ 以及由 $n$ 决定的 $t$. 算法的硬件实现电路大致分为区间压缩、查表、步 1、步 2 和扩展因子补偿 5 个模块，步 2 又根据 $i$ 的取值细分为两个流水组. 整条流水线被划分为 6 个流水组，每组内各级流水线结构相同.

模块 1：区间压缩，产生 $\phi$、$s_1\sim s_4$，含 1 级流水线；

模块 2：用 $\phi_H$ 查表获取 $\{d_i\}$、$\varepsilon$ 和 $K$，计算 $\phi_r$，并对 $x$、$y$ 初值进行变换，含 1 级流水线；

模块 3(步 1)：用系数 $\{d_i\}$ 控制旋转，每级处理 2 数(至多 1 个非零)，含 $\lceil t/2\rceil$ 级流水线；

模块 4-1(步 2-phase1)：$i<(m-1)/2$，用 $\phi_r$ 控制旋转，含 $\left\lceil\frac{m-3}{2}\right\rceil-t+1$ 级流水线；

模块 4-2(步 2-phase2)：$i\geq(m-1)/2$，合并相邻的两次迭代，用 $\phi_r$ 控制旋转，含 $\lceil(n+1-\lceil(m-1)/2\rceil)/2\rceil$ 级流水线；

模块 5：扩展因子补偿，含 $\lceil m/4\rceil$ 级流水线.

扩展因子补偿操作既可针对迭代初值，也可针对迭代终值，前者寄存 $\phi_r$ 的开销较大，后者寄存 $\{d_i\}$ 与 $K$ 的开销较大，二者性能相当，本文将扩展因子补偿操作置于最后. 2S-PCS 算法的实现流程如图 4 所示.

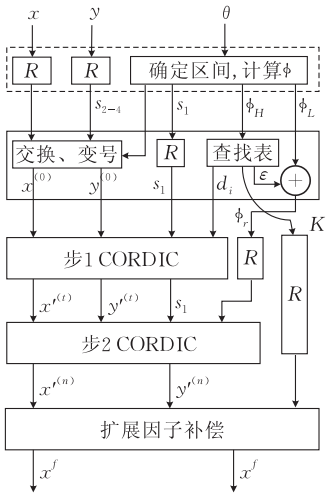


图 4 2S-PCS 算法的实现流程图

步 2-phase2 每级流水线处理 2 个迭代系数，合并相邻的两次迭代，并利用 CSA 优化累加操作. 合并后迭代表达式如下

$$\begin{aligned}\begin{pmatrix}x^{(i+2)}\\ y^{(i+2)}\end{pmatrix}&=\begin{pmatrix}1 & -d_{i+1}\cdot 2^{-(i+1)}\\ d_{i+1}\cdot 2^{-(i+1)} & 1\end{pmatrix}\begin{pmatrix}x^{(i+1)}\\ y^{(i+1)}\end{pmatrix}\\&=\begin{pmatrix}1 & -d_{i+1}\cdot 2^{-(i+1)}\\ d_{i+1}\cdot 2^{-(i+1)} & 1\end{pmatrix}\begin{pmatrix}1 & -d_i\cdot 2^{-i}\\ d_i\cdot 2^{-i} & 1\end{pmatrix}\begin{pmatrix}x^{(i)}\\ y^{(i)}\end{pmatrix}\\&=\begin{pmatrix}1-d_i\cdot d_{i+1}\cdot 2^{-(2i+1)} & -d_i\cdot 2^{-i}-d_{i+1}\cdot 2^{-(i+1)}\\ d_i\cdot 2^{-i}+d_{i+1}\cdot 2^{-(i+1)} & 1-d_i\cdot d_{i+1}\cdot 2^{-(2i+1)}\end{pmatrix}\begin{pmatrix}x^{(i)}\\ y^{(i)}\end{pmatrix}.\end{aligned}$$

当 $i\geq(m-1)/2$ 时，$2^{-(2i+1)}=0$，故上式可简化为

$$\begin{pmatrix}x^{(i+2)}\\ y^{(i+2)}\end{pmatrix}=\begin{pmatrix}1 & -(2d_i+d_{i+1})2^{-(i+1)}\\ (2d_i+d_{i+1})2^{-(i+1)} & 1\end{pmatrix}\begin{pmatrix}x^{(i)}\\ y^{(i)}\end{pmatrix}.$$

$x$、$y$ 通路各对应 3 个加数，可利用一级 CSA 和一级全加器实现，比常规 CORDIC 的两级全加延迟减小了近一半.

扩展因子补偿操作对应的流水线每级处理 4 位，至多含 2 个非零位. 这样可保证整条流水线每级至多累加 3 个操作数. 为降低复杂性，模块 3(步 1)每级仅处理 2 个迭代系数，而未将相邻的两次旋转合并. 若使其每级处理 4 个迭代系数，则控制复杂，会使步 1 成为瓶颈.

## 6 性能分析与误差模拟

2S-PCS 算法步 1 通过查表获取迭代系数，步 2 则直接将旋转角的二进制编码作为迭代系数，故整个过程不需要 $z$ 通道. 对于 I 类应用，目前可省略 $z$ 通道的算法只有 MVSF 算法，但其迭代次数随数据表示精度的增加呈指数增加. 当 $\phi$ 落在 SF 算法收敛域外时，MVSF 算法约需增加 $\lfloor\phi\cdot 2^t\rfloor+1$ 次旋转角为 $2^{-t}$ 的迭代. 以 $\phi=\pi/8$ 为例，当 $n=18$ 时，$\lfloor\phi\cdot 2^t\rfloor=25$；当 $n=21$ 时，$\lfloor\phi\cdot 2^t\rfloor=50$；当 $n=30$ 时，$\lfloor\phi\cdot 2^t\rfloor=402$. 而 2S-PCS 算法步 1 阶段的迭代次数仅为 $\lceil t/2\rceil$，可扩展性好.

另外 2S-PCS 算法通过约束步 1 非零系数的位置、对扩展因子进行基 4-Booth 编码、合并部分相邻的运算级等措施减少流水线级数. 以 $m=n=28$ 为

例，常规 CORDIC 算法需 $2+28+14=44$ 级流水线（包含为扩展收敛域而增加的 2 次 $\pi/4$ 旋转，并利用基 4-Booth 编码将扩展因子补偿操作转化为 14 级加法流水线），每级流水线延迟约为 1 个 28 位加法器延迟. 而 2S-PCS 算法仅需 $2+\lceil t/2\rceil+\left\lfloor\frac{m-3}{2}\right\rfloor-t+1+\lceil(n+1-\lceil(m-1)/2\rceil)/2\rceil+\lceil m/4\rceil=27$ 级流水线. 整条流水线每级至多累加 3 个操作数，CSA 的使用使每级流水线延迟仅略大于常规 CORDIC 算法，约为 1 位全加器和 1 个 $m$ 位加法器延迟之和，再加上简单的判断和选择逻辑的延迟. $x$、$y$ 通路利用固定连线＋简单的选择逻辑实现移位功能，充分利用了零迭代系数，又不需复杂的桶式移位器（Barrel Shift）. 这些方法使得 2S-PCS 算法相对于其他向量旋转算法具有明显的面积优势，若用加法器来衡量，则 2S-PCS 算法所需加法器数目为：3 个 $n+3$ 位常数加法器，1 个 $n-t+1$ 位加法器，$S_1$ 个 $m$ 位加法器和 $S_2$ 个 CSA（1 位全加器）. 其中

$$S_1=2\times\lceil t/2\rceil+2\times\left(\left\lceil\frac{m-3}{2}\right\rceil-t+1\right)+2\times\lceil(n+1-\lceil(m-1)/2\rceil)/2\rceil+2\times\lceil m/4\rceil,$$

$$S_2=m\times\left(\left\lceil\frac{m-3}{2}\right\rceil-t+1+\lceil(n+1-\lceil(m-1)/2\rceil)/2\rceil+\lceil m/4\rceil\right).$$

当 $m=n=28$ 时，总的加法器数目约为 54 个 28 位加法器和 588 个 1 位加法器（约合 21 个 28 位加法器），合 75 个 28 位加法器. 而常规 CORDIC 算法的旋转过程需 $(2+n)\times3=90$（含为扩展收敛域而增加的 2 次 $\pi/4$ 旋转）个 $m$ 位加法器，扩展因子补偿过程需 2 个 $m$ 位乘法器，若用基 4-Booth 乘法器，需 $m/2$ 个 $m$ 位加法器，整个常规 CORDIC 算法约需 $90+14=104$ 个 28 位加法器. 故 2S-PCS 算法可节省约 27.9% 的面积开销. 基于 Xilinx Virtex 系列 FPGA 以全流水方式分别实现了传统 CORDIC 旋转算法和 2S-PCS 算法，综合后发现：利用 IP 核实现 CORDIC 需 2160 个 slice，而 2S-PCS 算法仅需 1611 个 slice，二者均用加法代替乘法实现扩展因子补偿. 由理论分析可知，2S-PCS 算法的面积优势主要得益于 $z$ 通路的省略和流水线级数的压缩.

2S-PCS 算法的误差主要分为角度误差和 $x$、$y$ 通路误差. 步 1 过程的 $\varepsilon$ 因超出精度表示范围而产生圆整误差 $\phi_{r1}$，$\phi_H$ 相同的角对应的 $\phi_{r1}$ 相同，步 2 过程不产生剩余角误差. 区间压缩过程会产生圆整误差 $\phi_{r0}$，$n=28$（区间压缩前：1 位符号＋2 位整数＋28 位小数，区间压缩后：28 位无符号小数）时的剩余角误差分布如图 5 所示，纵坐标 $B=\lfloor-\log_2(|\phi_{rr}|)\rfloor$，$\phi_{rr}=\phi_{r0}+\phi_{r1}$. 可见，区间压缩会影响剩余角，但不总是副作用. 当输入角 $\theta\in(-\pi/4,\pi/4)$ 时，$\phi_{r0}=0$；$|\theta|\in(\pi/4,\pi/2)$ 时，$\phi_{r0}$ 与 $\phi_{r1}$ 同号概率较大，误差增大，而 $|\theta|\in(\pi/2,\pi)$ 时，$\phi_{r0}$ 与 $\phi_{r1}$ 异号概率较大，误差减小.

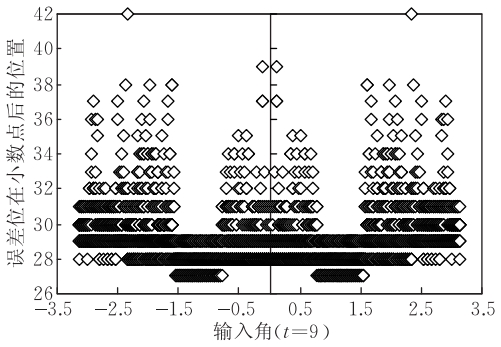


图 5　$n=28$ 时的角度误差

$x$、$y$ 通路的误差主要由剩余角和计算过程的圆整操作引起. $m=n=28$ 时，对单位向量的旋转进行模拟，用 1 位符号位＋1 位整数＋26 位小数表示 $x$，$y$，扩展因子也用 1 位符号（扩展因子恒正，符号位恒为 0，用补码表示便于 Booth 编码）＋1 位整数＋26 位小数表示. 设初始向量为 $(1,0)$，则 $x$ 通路的误差曲线如图 6 所示. 纵坐标表示 $B=\lfloor-\log_2(|x_f-x_r|)\rfloor$，$x_f$ 为计算值，$x_r$ 为准确值. 图 6(a) 对应 $t=\lceil(28-\log_2^6)/3\rceil=9$，图 6(b) 对应 $t=\lfloor(28-\log_2^6)/3\rfloor=8$. 图 7 为 $y$ 通路误差. 通过对比可发现，$t=7$ 或 9 时误差分布类似，最坏情况下的计算精度均可达到 22 位，因此实际应用中可取 $t=\lfloor(n-\log_2^6)/3\rfloor$，以减少查找表的入口数和流水线级数.

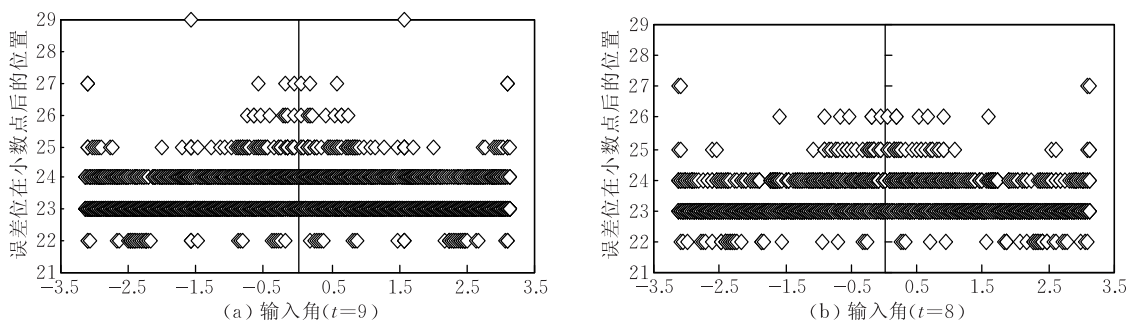


图 6　$m=n=28$（26 位小数）时 $x$ 通路的误差

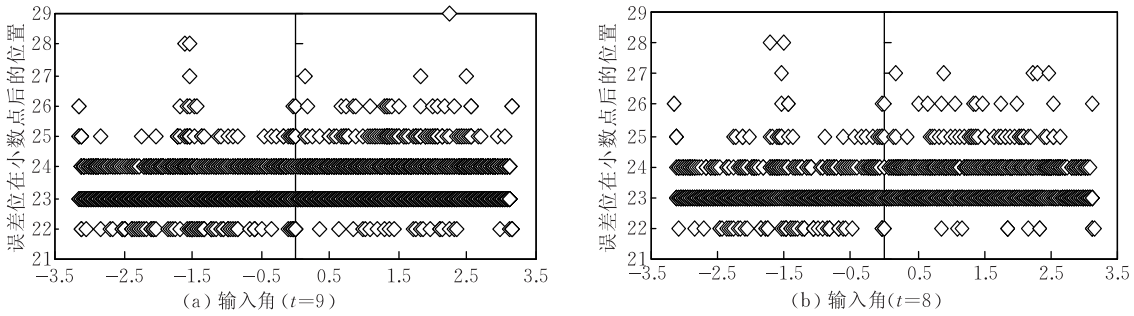


图 7 $m=n=28$(26 位小数)时 $y$ 通路的误差

实验表明:2S-PCS 算法具有较高的计算精度. 常规 CORDIC 算法需使用$(n+\log_2 n+2)$位数据通路[12]才能达到 $n$ 位精度,用 28 位数据通路,只能达到约 21 位精度,去掉小数点前两位,精度只能达到小数点后 19 位,比 2S-PCS 算法的精度(22 位小数)低 3 位. 2S-PCS 算法的高精度主要得益于短流水线,且利用 CSA 将 3 个加数化为 2 个,每级流水线只引发一次圆整误差. 本例中,整条 27 级流水线的误差限约为$\frac{1}{2}\times 27\times 2^{-26}$,精度降低 4 位左右. 另外,$d_i$为 0 的级不存在圆整误差,故 2S-PCS 算法的平均精度要比最低精度高 1～2 位.

## 7 总 结

本文提出一种新的 CORDIC 算法和体系结构,减少迭代次数的同时保持了常规 CORDIC 算法扩展因子易计算与补偿的特点,实现了旋转角位于$[-\pi,\pi]$的向量旋转操作. 2S-PCS 算法根据数据通路 $x$,$y$ 以及旋转角 $\phi$ 的表示精度确定流水线级数和组织方式. 步 1 的迭代系数和扩展因子由查表获得,步 2 迭代系数直接取目标转角的二进制编码,扩展因子恒为 1,因此整个旋转过程不需 $z$ 通路和扩展因子计算通路,大大减小了电路的面积开销. 通过带约束的角度分解算法、基 4-Booth 编码和迭代合并等措施,有效减少了流水线级数. 2S-PCS 算法在迭代过程中对超出计算精度的表达式预先进行化简,并采用 CSA 累加多个操作数,有效地减少了流水线延迟. 算法克服了传统全字段查表法可扩展性差的问题,随着数据表示精度的增加,查找表入口数增加缓慢,可扩展性好. 理论分析与误差模拟表明,2S-PCS 算法在保持高精度的同时具有明显的面积和延时优势,实用性强. 采用 28 位数据通路时,与常规 CORDIC 算法相比,2S-PCS 算法的流水线级数减少约 38%,面积减少约 27.9%,精度提高 3 位左右,显示了算法的优良性能.

## 参 考 文 献

[1] Volder J E. The CORDIC trigonometric computing technique. IRE Transactions Electronic Computing, 1959, 8(9): 330-334

[2] Hu Y H, Naganathan S. An angle recoding method for CORDIC algorithm implementation. IEEE Transactions on Computers, 1993, 42(1): 99-102

[3] Wu C S, Wu A Y. Modified vector rotational CORDIC (MVR-CORDIC) algorithm and its application to FFT//Proceedings of the IEEE International Symposium Circuits and Systems. Geneva, 2000: 529-532

[4] Wu C S, Wu A Y. A novel rotational VLSI architecture based on Extended Elementary-angle Set CORDIC algorithm//Proceedings of the 2nd IEEE Asia Pacific Conference on ASICs. Cheju, Korea, 2000: 111-114

[5] Lin Z-X, Wu A-Y. Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for scaling-free high-performance rotational operations//Proceedings of the IEEE International Conference. Acoustics, Speech, Signal Processing (vol. 2). Hong Kong, China, 2003: 653-656

[6] Chih Jen-Chuan, Chen Sau-Gee. Fast CORDIC algorithm based on a new recoding scheme for rotation angles and variable scale factors. Journal of VLSI Signal Processing, 2003, 33(1): 19-29

[7] Li C C, Chen S G. A radix-4 redundant CORDIC algorithm with fast on-line variable scale factor compensation//Proceedings of the 1997 IEEE International Conference on Acoustic, Speech and Signal Processing. Munich, Germany, 1997: 639-642

[8] Li C C, Chen S G. New redundant CORDIC algorithm with variable scale factor compensations//Proceedings of the 1996 IEEE International Symposium Circuits and Systems. Atlanta, USA, 1996: 264-267

[9] Koushik Maharatna, Swapna Banerjee. Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture. IEEE Transactions on Circuits and Systems for Video Technology, 2005, 15(11): 1463-1474

[10] Maharatna K, Dhar A S, Banerjee S. A VLSI array architecture for realization of DFT, DHT, DCT and DST. Signal Processing, 2001, 81(9): 1813-1822

[11] Mathematics Teaching and Research Office of Tongji Univer-

sity. Advanced Mathematics(Volume I). 4th Edition. Beijing: China Higher Education Press, 1996: 177(in Chinese)
(同济大学数学教研室. 高等数学(上册). 第 4 版. 北京：高等教育出版社，1996：177)

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

[13] Lei Yuan-Wu, Zhou Jie et al. Research of the parallel CORDIC algorithm and its implementation in FPGA. Computer Engineering and Science, 2008, 30(8): 75-78(in Chinese)
(雷元武，周杰等. 并行 CORDIC 算法的研究及 FPGA 实现. 计算机工程与科学，2008，30(8)：75-78)

**附录.** CHAR 算法证明.

显见算法满足约束条件 2)，只需证明 $|\varepsilon|<\alpha(t)$ 即可. 首先用归纳法证明 $|\phi(j)|<\alpha(2j)$，$1\le j\le\left\lfloor\frac{t}{2}\right\rfloor$. 已知 $|\phi(0)|\le\alpha(0)$，即 $j=0$ 时 $|\phi(j)|\le\alpha(2j)$ 成立. 设 $j=k$ 时 $|\phi(k)|\le\alpha(2k)$，则

(1) $\frac{\alpha(2k)+\alpha(2k+1)}{2}\le|\phi(k)|<\alpha(2k)$ 时，考虑到 $\alpha(i)<2\alpha(i+1)$(三角函数性质)，有

$$0\le\alpha(2k)-|\phi(k)|\le\frac{\alpha(2k)-\alpha(2k+1)}{2}<\frac{\alpha(2k+1)}{2}<\alpha(2k+2),$$

$$|\phi(k+1)|=|\phi(k+1)_2|=|\alpha(2k)-|\phi(k)||<\alpha(2k+2).$$

(2) $\frac{\alpha(2k+1)}{2}\le|\phi(k)|<\frac{\alpha(2k)+\alpha(2k+1)}{2}$ 时，

$$-\frac{\alpha(2k+1)}{2}\le|\phi(k)|-\alpha(2k+1)<\frac{\alpha(2k)-\alpha(2k+1)}{2}<\frac{\alpha(2k+1)}{2},$$

故 $|\phi(k+1)|=|\phi(k+1)_1|\le\frac{\alpha(2k+1)}{2}<\alpha(2k+2)$.

(3) $|\phi(k)|<\frac{\alpha(2k+1)}{2}$ 时，$|\phi(k+1)|=|\phi(k)|<\frac{\alpha(2k+1)}{2}<\alpha(2k+2)$.

综合上述 3 种情况，可得 $|\phi(k+1)|<\alpha(2k+2)$. 由推导过程可知 $1\le j\le\left\lfloor\frac{t}{2}\right\rfloor$ 时，$|\phi(j)|$ 与 $\alpha(2j)$ 为真小于关系，不可能相等. 故原式得证.

For 循环结束后，若 $t$ 为偶数定有 $|\varepsilon|=\left|\phi\left(\frac{t}{2}\right)\right|<\alpha(t)$；若 $t$ 为奇数定有 $\left|\phi\left(\frac{t-1}{2}\right)\right|<\alpha(t-1)$. 当 $t$ 为奇数且 $\left|\phi\left(\frac{t-1}{2}\right)\right|\le\alpha(t)$ 时，需继续计算最后一个独立成组的系数. 此时

若 $\left|\phi\left(\frac{t-1}{2}\right)\right|<\frac{\alpha(t-1)}{2}$，则 $|\varepsilon|=\left|\phi\left(\frac{t+1}{2}\right)\right|=\left|\phi\left(\frac{t-1}{2}\right)\right|<\frac{\alpha(t-1)}{2}<\alpha(t)$；

若 $\frac{\alpha(t-1)}{2}\le\left|\phi\left(\frac{t-1}{2}\right)\right|<\alpha(t-1)$，则 $|\varepsilon|=\left|\phi\left(\frac{t+1}{2}\right)\right|=\alpha(t-1)-\left|\phi\left(\frac{t-1}{2}\right)\right|<\frac{\alpha(t-1)}{2}<\alpha(t)$.

故无论 $t$ 为奇数或偶数，算法均可保证误差 $|\varepsilon|<\alpha(t)$.

**MOU Sheng-Mei**, born in 1979, Ph. D., lecturer. Her main research interests include computer architecture, hardware acceleration of algorithms, embedded system design, etc.

**YANG Xiao-Dong**, born in 1936, professor, Ph. D. supervisor. His research interests include computer architecture, distributed and parallel processing and RAS, etc.

**Background**

CORDIC algorithm has been widely used in digital signal processing, scientific computing and other applications for efficient implementation of vector rotation operations in hardware. Lookup table is usually used to provide coefficients (rotation direction) and scale factors in applications with predictable rotation angles; however, it's seldom used in those applications with flexible rotation angles due to its poor scalability. Instead, coefficients must be computed dynamically through the angle approximation datapath ($z$ datapath). Conventional CORDIC are attractive with constant scale factors, but are slow as a result of its large iteration numbers. Some other modified algorithms bypass unnecessary iterations by permitting coefficient $d_i$ to equal zero but bring the side effect of variable scale factors, which have to be computed dynamically. In addition, these algorithms can't decrease the length of pipeline efficiently because of unpredictable zero positions.

Stuck in such dilemma, illuminated by scaling-free CORDIC algorithm, the authors propose 2S-PCS algorithm. It integrates conventional CORDIC with scaling-free CORDIC, which makes it possess predictable scale factors as conventional CORDIC, easily gotten coefficients, and reduced non-zero coefficients on constrained positions. It's a great deal of area conservation without the cost of computing coefficients and scale factors. The short pipeline is also beneficial to high accuracy. Experiments and theoretical analysis testify that 2S-PCS algorithm excels in vector rotation with less area consumption, short latency and high accuracy.