文章编号:1001 - 893X(2011)08 - 0085 - 05

CORDIC 算法的一种补码实现结构设计*

孙 学

(中国西南电子技术研究所,成都 610036)

摘 要:根据 CORDIC 算法原理,分析了该算法角度旋转范围缺陷,提出 360°覆盖的角度旋转算法结构;推导出利用补码实现 CORDIC 算法的迭代运算单元结构,并根据该补码运算原理设计了 CORDIC 补码迭代运算单元和方向向量发生器的实现结构。

关键词:坐标旋转数字计算机;方向向量;补码电路

中图分类号:TN919.3;TP301 **文献标识码**:A doi:10.3969/j.issn.1001 - 893x.2011.08.018

Circuit Design of Complementary Form CORDIC Algorithm

SUN Xue

(Southwest China Institute of Electronic Technology, Chengdu 610036, China)

Abstract: This paper analyses the limitation of angle rotation range in CORDIC (Coordinate Rotation Digital Computer) algorithm according to its principle, and proposes an angle rotation structure covering 360 degree, and designs the pipeline circuit architecture of direction vector generator. The mathematical deduction of CORDIC operator is given based on complementary form shifting and the circuit design of CORDIC iterations.

Key words: CORDIC; direction vector; complementary circuit

1 引 言

CORDIC 是 Coordinate Rotation Digital Computer (坐标旋转数字计算机)的首字母缩写,该数值逼近迭代算法由 J.E.Volder 于 1959 年提出[1]。其主要优点在于用简单的移位和加减运算取代查三角函数表和乘法运算,实现二维矢量的旋转运算,特别适合于大规模集成电路的实现。另外,CORDIC 算法用于估算平方根、正余弦、指对数等初等函数的特点[2],被广泛应用于导航解算、频偏估计、调制解调和频率合成等矢量运算场合[3-5]。

本文第 2 节基于 CORDIC 算法原理分析了该算法实现角度旋转存在覆盖范围缺陷,提出一种 360° 角度覆盖的 CORDIC 旋转算法结构。第 3 节基于该算法结构,创新性地推导出 CORDIC 迭代运算单元的一种补码实现结构,是本文的描述重点。

* 收稿日期:2011 - 03 - 01;修回日期:2011 - 06 - 10

2 CORDIC 算法

2.1 CORDIC 算法原理[1]

通过旋转一系列小角度的以偏摆逼近所需的角度、开方以及反三角函数等复杂运算逻辑,复数 $P = x + jy$ 旋转 θ 角度得到 Q 的过程就是复数 P 与复指数 $e^{j\theta}$ 进行复乘得到 Q 的过程。如果旋转角度 θ 可以分解成 N 个小角度 ϕ_i 之和,那么旋转角度 θ 就等效于旋转这 N 个小角度 ϕ_i。

$$Q = Pe^{j\theta} = Pe^{j(\phi_0 \pm \phi_1 \pm \phi_2 \pm \cdots \pm \phi_{N-1})} = Pe^{j\phi_0}e^{j\phi_1}\cdots e^{j\phi_{N-1}} \tag{1}$$

其角度 θ 分解方法为

$$\theta = \phi_0 \pm \phi_1 \pm \phi_2 \pm \cdots \pm \phi_{N-1} = \sum_{i=0}^{N-1} \delta_i \phi_i$$

$$(\theta \cdot \phi_i \geqslant 0; \delta_i = \pm 1; \delta_0 = 1)$$

Q 的实部和虚部表达式为

$$\begin{cases} Q_{re} = x\cos\theta - y\sin\theta = (x - y\mathrm{tg}\theta)\cos\theta \\ Q_{im} = y\cos\theta + x\sin\theta = (y + x\mathrm{tg}\theta)\cos\theta \end{cases} \quad (2)$$

令 $x = R_0, y = I_0, \phi_i = \mathrm{arctg}\ 2^{-i}$,则 $R_i + \mathrm{j}I_i$ 旋转 $\delta_i\phi_i$ 角度得到 $R_{i+1} + \mathrm{j}I_{i+1}$ 的实部和虚部分别为

$$\begin{cases} R_{i+1} = (R_i - I_i\delta_i 2^{-i})\cos\phi_i \\ I_{i+1} = (I_i + R_i\delta_i 2^{-i})\cos\phi_i \end{cases} \quad (3)$$

若令 $x = R_0 = \hat{R}_0, y = I_0 = \hat{I}_0$,并且令 $\hat{R}_i$ 和 $\hat{I}_i$ 的迭代运算关系为

$$\begin{cases} \hat{R}_{i+1} = \hat{R}_i - \hat{I}_i\delta_i 2^{-i} \\ \hat{I}_{i+1} = \hat{I}_i + \hat{R}_i\delta_i 2^{-i} \end{cases}, i = 0, 1, \cdots, N-1 \quad (4)$$

那么经过 N 次迭代可得:

$$\begin{cases} R_N = \hat{R}_N \prod_{i=0}^{N-1} \cos\phi_i = Q_{re} \\ I_N = \hat{I}_N \prod_{i=0}^{N-1} \cos\phi_i = Q_{im} \end{cases} \quad (5)$$

这就是说,计算出 $\hat{R}_N + \mathrm{j}\hat{I}_N$ 后,再乘以 $\prod_{i=0}^{N-1}\cos\phi_i$ 得到的结果就是复数 P 旋转角度 θ 后得到的 Q。

$$\prod_{i=0}^{N-1}\cos\phi_i = \frac{1}{\prod_{i=0}^{N-1}\sqrt{1 + (2^{-i})^2}} = \frac{1}{K} \quad (6)$$

式中,$K \approx \begin{cases} 1.646\ 8, N = 15 \\ 1.647, N \text{ 足够大} \end{cases}$。

这说明只需要按迭代公式(5)进行简单的移位和加减运算迭代计算出 $\hat{R}_N + \mathrm{j}\hat{I}_N$ 后,经过补偿因子 K 的 1.647 倍幅度校正即可计算出复数 P 旋转角度 θ 后得到 Q 的近似值,其实现结构如图 1 所示。

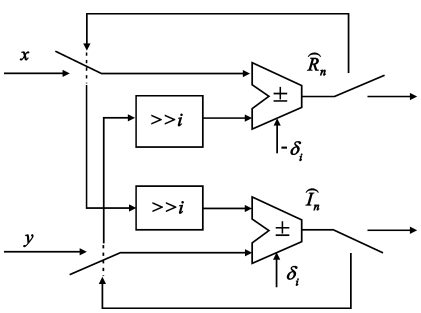

图 1 CORDIC 算法迭代结构框图

Fig.1 Circuit of original form CORDIC iteration operator

2.2 角度旋转范围缺陷及补偿设计

由于 $\phi_i = \mathrm{arctg}\ 2^{-i}$ 在 $i = 0, 1, \cdots, N-1$ 时小于等于 $\pi/4$ 且单调递减,则 $\theta = \sum_{i=0}^{N-1}\delta_i\phi_i$($N$ 趋于无穷大)收敛,下面证明 θ 收敛范围。

$$\mathrm{arctg}\ x = \int_0^x \frac{1}{1+t^2}\mathrm{d}t = \int_0^x \left(\sum_{n=0}^{\infty}(-1)^n t^{2n}\right)\mathrm{d}t = \sum_{n=0}^{\infty}(-1)^n \int_0^x t^{2n}\mathrm{d}t = \sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{2n+1}$$

其中,x 的范围为$[-1, +1]$,则

$$\theta = \sum_{i=0}^{N-1}\delta_i \mathrm{arctg}\ 2^{-i} = \sum_{i=0}^{N-1}\delta_i \sum_{n=0}^{\infty}(-1)^n \frac{(2^{-i})^{2n+1}}{2n+1} = \sum_{n=0}^{\infty}\sum_{i=0}^{N-1}\delta_i(-1)^n \frac{(2^{-i})^{2n+1}}{2n+1} \quad (7)$$

当 $\delta_i \equiv \pm 1$ 时,θ 取最大(小)值。令 $\delta_i \equiv 1$,则:

$$\theta = \sum_{i=0}^{N-1}\mathrm{arctg}\ 2^{-i} = \sum_{n=0}^{\infty}\sum_{i=0}^{N-1}(-1)^n \frac{(2^{-i})^{2n+1}}{2n+1} = \sum_{n=0}^{\infty}(-1)^n \frac{1 - \left(\left(\frac{1}{2}\right)^{2n+1}\right)^N}{(2n+1)\left(1 - \left(\frac{1}{2}\right)^{2n+1}\right)} = \sum_{n=0}^{\infty}(-1)^n \frac{1}{(2n+1)}\sum_{i=0}^{N-1}\left(\left(\frac{1}{2}\right)^{2n+1}\right)^i = \sum_{n=0}^{\infty}(-1)^n u(n) \quad (8)$$

其中 $u(n) = \frac{1}{(2n+1)}\sum_{i=0}^{N-1}\left(\left(\frac{1}{2}\right)^{2n+1}\right)^i$,显然 $u(n) \geqslant u(n+1)$; $\lim_{n\to\infty} u(n) = 0$。由莱布尼兹交错级数收敛性判别法可知,式(8)级数收敛,且:

$$\sum_{n=0}^{\infty}(-1)^n u(n) < (u(0) = 1 \approx 115°) \quad (9)$$

同理可证,$\sum_{n=0}^{\infty}(-1)^n u(n)$ 的下限大于 $-115°$。仿真计算式(8)中 θ 的取值范围为$(-99.880°, +99.880°)$($N = 16$);$(-99.883°, +99.883°)$(N 足够大)。

因此,用 CORDIC 算法实现数据的相位旋转时,需要扩大 CORDIC 算法的角度旋转范围,从$(-99.883°, +99.883°)$扩展到$[-180°, 180°]$,在实际工程应用中,通常把$[-180°, 180°]$表达为$[0°, 360°]$或$[-360°, 0°]$的角度旋转,设计如下。

$$\theta_1 = \theta + \left(-\frac{3\pi}{2}\right) \in (-369.883°, +9.883°)$$

$$\theta_2 = \theta + \left(\frac{3\pi}{2}\right) \in (-9.883°, +369.883°) \quad (10)$$

令 $\hat{\theta} = \theta + \sum_{m=1}^{6}\left(\frac{\pi}{4} \times \hat{\delta}_m\right)$,则 $\hat{\theta} \in (-369.883°, +369.883°)$,覆盖 360°的角度旋转范围。取 $N = 16$

时,则

$$\hat{\theta} \approx \delta_i \sum_{i=0}^{15} \operatorname{arctg} 2^{-i} + \sum_{m=1}^{6} \left(\frac{\pi}{4} \times \hat{\delta}_m\right) \tag{11}$$

该方法是在 CORDIC 移位序列前加上 6 个 $\phi_i = \operatorname{arctg} 2^{-i}(i=0)$迭代,在 270°范围内实现旋转步进为 45°的 $\hat{\theta}$ 粗定位,在此基础上通过不断减小角度旋转步进摇摆逼近 $\hat{\theta}$,实现 $\hat{\theta}$ 的精确定位。式(11)设计旋转角度粗定位方法,导致复数 P 每旋转 45°后幅度放大 1.414 倍,使得式(5)中校正因子 K 放大 8 倍。在实现式(11)的 CORDIC 处理器时,为了减小旋转过程中对复数 P 实部和虚部表达范围的影响,在这 6 个迭代的每两级旋转后,实部和虚部各右移一位,缩小一倍以免数值溢出。

3　CORDIC 补码实现结构设计

3.1　CORDIC 的原码实现原理

根据 CORDIC 算法迭代结构框图,设计其原码实现结构,如图 2 所示。

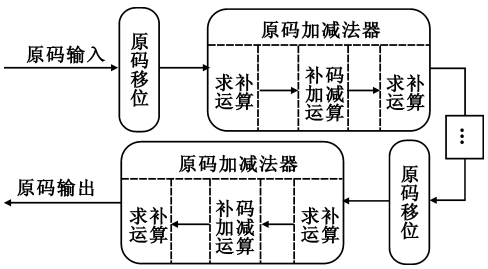

图 2　CORDIC 的原码实现原理

Fig.2 Principle of original form CORDIC

CORDIC 算法的原码实现结构优点在于原码移位简单,缺点在于补码方式实现二进制加减法前后,都需要进行求补运算,原码实现结构中每一级迭代运算都需要 4 个补码器并导致 2 级求补运算时延。采用流水线结构实现式(11)的 CORDIC 需要 88 个补码器,导致 44 级求补运算时延,不但会导致器件资源的大量消耗,在高实时矢量运算场合还将带来严重的时延问题。为此,本文设计了一种补码实现结构的 CORDIC 处理器,能够有效解决以上技术问题。

3.2　CORDIC 的补码实现原理

补码实现 CORDIC 处理器的基本思想是:原码数据在输入 CORDIC 运算器之前进行补码运算,在 CORDIC 运算器的运算过程中全部采用补码运算,CORDIC 运算结果进行求补运算,输出原码数据,如图 3 所示。

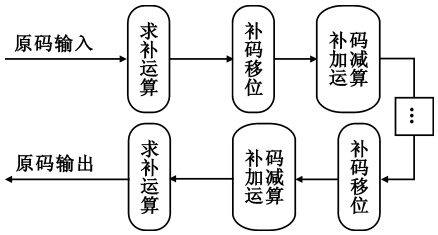

图 3　CORDIC 的补码实现原理

Fig.3 Principle of complementary form CORDIC

该实现方法的关键在于补码移位算法和 CORDIC 补码迭代单元设计。

3.3　补码移位器设计

假设输入的整数 X 的原码数据为

$$[X]_{原码} = D_o = B_S B_N B_{N-1} \cdots B_i \cdots B_1 B_0 \tag{12}$$

其中,B_S 为符号位,“1”表示负数,“0”表示正数,$0 \leqslant i \leqslant N$ 表示数据的数值位。需要说明的是:十进制 0 的原码表示为符号位 $B_S = 0$,即只有正“零”,不允许负“零”的出现。

定义负整数 X 的反码为

$$[X]_{反码} = D_n = B_S \bar{B}_N \bar{B}_{N-1} \cdots \bar{B}_i \cdots \bar{B}_1 \bar{B}_0 \tag{13}$$

即符号位不变,数值位进行取反运算;正整数 X 的反码为它的原码本身。

定义数据的补码为

$$[X]_{补码} = \begin{Bmatrix} X, & 0 \leqslant X < 2^n \\ 2^{n+1} + X, & -2^n \leqslant X < 0 \end{Bmatrix} \mathrm{MOD}\ 2^{n+1} \tag{14}$$

式中,n 为二进制整数数值位的位数。由定义可以推出,补码的求补运算结果为数据的原码。为了得到一个数的补码表示,当然可以通过补码的定义求得,但更简便的办法如下。

(1) $B_S = 0$ 时,正整数数据的补码等于它的原码本身:

$$D_c = B_S B'_N B'_{N-1} \cdots B'_i \cdots B'_1 B'_0 = B_S B_N B_{N-1} \cdots B_i \cdots B_1 B_0 \tag{15}$$

(2) $B_S = 1$ 时,负整数数据的补码等于它的反码加“1”:

$$D_c = B_S B'_N B'_{N-1} \cdots B'_i \cdots B'_1 B'_0 = B_S \bar{B}_N \bar{B}_{N-1} \cdots \bar{B}_i \cdots \bar{B}_1 \bar{B}_0 + 1 \tag{16}$$

由于正数的补码等于它的原码本身,所以正数的补码移位和它的原码移位规律相同,真值除以 2^i 的补码实现和真值除以 2^i 的原码实现方式相同,不用讨论,现在要研究的是负数的补码实现真值除以 2^i 运算规律。负数的原码表示为

$$D_0 = 1, B_N B_{N-1} \cdots B_i \cdots B_1 B_0 \tag{17}$$

其补码为

$$D_1 = 1, B'_N B'_{N-1} \cdots B'_i \cdots B'_1 B'_0 \tag{18}$$

则下式成立：

$$1, B'_N B'_{N-1} \cdots B'_i \cdots B'_1 B'_0 = 1, \bar{B}_N \bar{B}_{N-1} \cdots \bar{B}_i \cdots \bar{B}_1 \bar{B}_0 + 1 \tag{19}$$

D_1右移 i 位后变为

$$D_2 = 1, 11 \cdots 11, B'_N B'_{N-1} \cdots B'_{i+1} B'_i \tag{20}$$

D_0右移 i 位后变为

$$D_3 = 1, 00 \cdots 00, B_N B_{N-1} \cdots B_i \tag{21}$$

把 D_1右移 i 位后的 D_2经过一定的处理得到 D_4，使得 D_4的真值与 D_3的真值相等，这就达到了补码实现真值除以 2^i 的目的。

D_3的补码为

$$D_5 = 1, 11 \cdots 11, \bar{B}_N \bar{B}_{N-1} \cdots \bar{B}_i + 1 \tag{22}$$

$\bar{B}_{i-1}$and $\bar{B}_{i-2}$and$\cdots$and $\bar{B}_1$and $\bar{B}_0 = 1$ 时，则 $D_2 = D_5$，即 $B'_{i-1}$and $B'_{i-2}$and$\cdots$ and $B'_1$and $B'_0 = 0$ 时，补码右移 i 位就达到了真值除以 2^i 的目的。

$\bar{B}_{i-1}$or $\bar{B}_{i-2}$or$\cdots$or $\bar{B}_1$or $\bar{B}_0 = 0$ 时，则 $D_2 + 1 = D_5$，即 B'_{i-1}or B'_{i-2}or$\cdots$or B'_1or $B'_0 = 1$ 时，补码右移 i 位的结果再加上“1”才实现真值除以 2^i 的功能。

综上所述，补码实现真值除以 2^i 功能的实现主要是靠补码右移 i 位来实现，其实现方法是：

(1) $B_S = 0$，则真值除以 2^i 的补码实现和真值除以 2^i 的原码实现方式相同，都是直接右移 i 位，高位补“0”来实现真值除以 2^i；

(2) $B_S = 1$，如果 B'_{i-1}or B'_{i-2}or$\cdots$or B'_1 or $B'_0 = 1$ 为假，真值除以 2^i 的补码实现方式是右移 i 位，高位补“1”；如果 B'_{i-1}or B'_{i-2}or$\cdots$or B'_1or $B'_0 = 1$ 为真，真值除以 2^i 的补码实现方式是右移 i 位，高位补“1”的基础上再加上“1”。

综合以上两种方法得到二进制数的真值除以 2^i 的补码表达式为

$$[D_{原}/2^i]_{补} = (B_S, B_S B_S \cdots B_S B_S, B'_N B'_{N-1} \cdots B'_{i+1} B'_i) + \{B_S \text{and}(B'_{i-1} \text{or} B'_{i-2} \text{or} \cdots \text{or} B'_1 \text{or} B'_0)\} \tag{23}$$

式(23)的实现结构如图 4 所示。

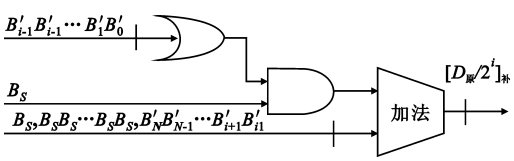

图 4 补码移位器结构设计

Fig.4 Complementary form shifting

3.4 方向向量发生器设计

设旋转因子 W_N^k 相位旋转通过 N 次 CORDIC 迭代实现，则方向向量 δ_i 集合中的元素个数为 N。实现该算法的一种运算结构是预先计算好 δ_i 后存储在 ROM 中，这种方案仅适合固定角度的旋转运算，不可能为每一种旋转角度取值进行预先的旋转向量计算。为此，本文提出方向向量 δ_i 实时发生器的设计方案解决了该问题。

旋转因子 W_N^k 旋转的相位角度为 $-\frac{2\pi}{N}k$，令收敛的误差角度为 $\tilde{\theta}_i$：

$$\tilde{\theta}_{i+1} = \tilde{\theta}_i - \delta_i \phi_i \approx 0 = \left(-\frac{2\pi}{N}k - \sum_{j=0}^{i-1}(\delta_j \text{arctg}\, 2^{-j})\right) - \delta_i \text{arctg}\, 2^{-i} \tag{24}$$

则 $\tilde{\theta}_i$ 的符号位就是方向向量 δ_i 的取值。比较 $\phi'_i = \frac{N}{2\pi}(\text{arctg}\, 2^{-i})$ 和 $\theta'_i = -k$ 的大小计算角度误差，其结果的最高位 MSB_i 就是方向向量 δ_i 的取值，这样的硬件设计更为简单，δ_i 为正时角度误差运算为减法，δ_i 为负时该运算为加法，式(11)的方向向量 δ_i 发生器实现结构如图 5 所示。

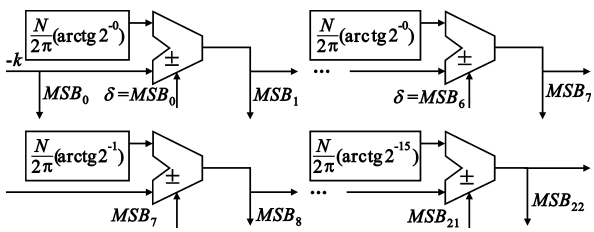

图 5 方向向量 δ_i 发生器

Fig.5 Direction vector generator

3.5 CORDIC 补码迭代单元设计

根据$[x \pm y]_{补码} = [x]_{补码} \pm [y]_{补码}$，也就是说，$x \pm y$ 真值的补码等于$[x]_{补码}$和$[y]_{补码}$二进制加减法的运算结果。结合图 3 的 CORDIC 补码实现原理，使用补码加减法器实现 CORDIC 迭代运算过程中的加减法，设计式(4)的补码迭代运算单元。

$$[\hat{R}_{i+1}]_{补码} = [\hat{R}_i - \hat{I}_i \delta_i 2^{-i}]_{补码} = [\hat{R}_i]_{补码}(\delta_i)[\hat{I}_i 2^{-i}]_{补码} \tag{25}$$

其中，当 $\delta_i = +1$ 时，(δ_i)表示减法运算；当 $\delta_i = -1$ 时，(δ_i)表示加法运算。代入式(23)，则：

$$
\begin{aligned}
[\widehat{R}_{i+1}]_{补码} = {} & [\widehat{R}_i]_{补码}(\delta_i)\{(\widehat{I}_{iB_S}, \widehat{I}_{iB_S}\widehat{I}_{iB_S}\cdots \\
& \widehat{I}_{iB_S}\widehat{I}_{iB_S}, \widehat{I}_{iB'_N}\widehat{I}_{iB'_{N-1}}\cdots\widehat{I}_{iB'_{i+1}}\widehat{I}_{iB'_i}) + \\
& [\widehat{I}_{iB_S}\mathrm{and}(\widehat{I}_{iB'_{i-1}}\mathrm{or}\widehat{I}_{iB'_{i-2}}\mathrm{or}\cdots\mathrm{or}\widehat{I}_{iB'_1}\mathrm{or}\widehat{I}_{iB'_0})]\} = \\
& [\widehat{R}_i]_{补码}(\delta_i)(\widehat{I}_{iB_S}, \widehat{I}_{iB_S}\widehat{I}_{iB_S}\cdots \\
& \widehat{I}_{iB_S}\widehat{I}_{iB_S}, \widehat{I}_{iB'_N}\widehat{I}_{iB'_{N-1}}\cdots\widehat{I}_{iB'_{i+1}}\widehat{I}_{iB'_i})(\delta_i) \\
& [\widehat{I}_{iB_S}\mathrm{and}(\widehat{I}_{iB'_{i-1}}\mathrm{or}\widehat{I}_{iB'_{i-2}}\mathrm{or}\cdots\mathrm{or}\widehat{I}_{iB'_1}\mathrm{or}\widehat{I}_{iB'_0})] = \\
& [\widehat{R}_i]_{补码}(MSB_i)(\widehat{I}_{iB_S}, \widehat{I}_{iB_S}\widehat{I}_{iB_S}\cdots \\
& \widehat{I}_{iB_S}\widehat{I}_{iB_S}, \widehat{I}_{iB'_N}\widehat{I}_{iB'_{N-1}}\cdots\widehat{I}_{iB'_{i+1}}\widehat{I}_{iB'_i})(MSB_i) \\
& [\widehat{I}_{iB_S}\mathrm{and}(\widehat{I}_{iB'_{i-1}}\mathrm{or}\widehat{I}_{iB'_{i-2}}\mathrm{or}\cdots\mathrm{or}\widehat{I}_{iB'_1}\mathrm{or}\widehat{I}_{iB'_0})]
\end{aligned}
\tag{26}
$$

同理,可得

$$
\begin{aligned}
& [\widehat{I}_{i+1}]_{补码} = [\widehat{I}_i]_{补码}(\mathrm{not}(MSB_i)) \\
& (\widehat{R}_{iB_S}, \widehat{R}_{iB_S}\widehat{R}_{iB_S}\cdots\widehat{R}_{iB_S}\widehat{R}_{iB_S}, \widehat{R}_{iB'_N}\widehat{R}_{iB'_{N-1}}\cdots\widehat{R}_{iB'_{i+1}}\widehat{R}_{iB'_i}) \\
& (\mathrm{not}(MSB_i))[\widehat{R}_{iB_S}\ \mathrm{and}\ (\widehat{R}_{iB'_{i-1}}\ \mathrm{or}\widehat{R}_{iB'_{i-2}}\ \mathrm{or}\cdots\mathrm{or}\widehat{R}_{iB'_1} \\
& \mathrm{or}\widehat{R}_{iB'_0})]
\end{aligned}
\tag{27}
$$

因此,设计 CORDIC 补码迭代运算单元的实现结构如图 6 所示。

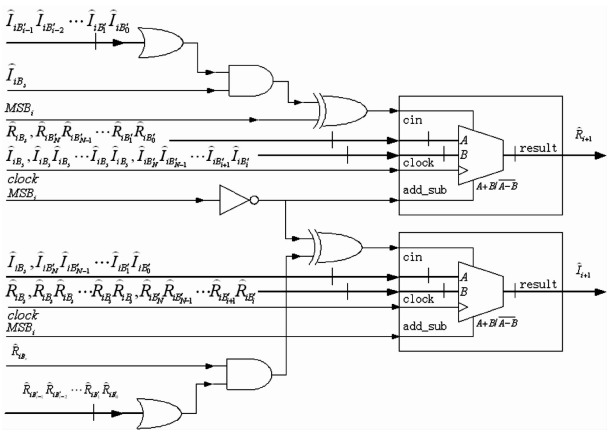

图 6　CORDIC 补码运算单元实现结构

Fig.6 CORDIC iteration operator based on complementary form shifting

4　小　结

CORDIC 作为一种计算向量旋转的迭代算法,算法结构简单,易于并行化处理和 VLSI 实现,因而在实时信号处理方面有广泛的应用前景。本文提出一种补码实现 CORDIC 的流水线电路结构,与其源码实现结构比较,省去每级迭代运算单元的 2 次求补运算过程,运算时延减少了一半,有效缓解了 CORDIC 算法固有收敛速度与高实时矢量运算场合低时延要求的矛盾。该设计可以作为 CORDIC 算法实现的一种技术参考。

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 [C]//Proceedings of American Federation of Information Processing Societies Spring Joint Computer Conference. New York: ACM, 1971: 379 – 385.

[3] Stephan W Mondwurf. Benefits of the CORDIC Algorithm in a Versatile Cofdm Modulator/Demodulator Design[C]//Proceedings of the Fourth IEEE International Caracas Conference on Devices, Circuits and Systems. Aruba: IEEE, 2002: 117 – 119.

[4] Despain A M. Fourier Transform Computers Using CORDIC Iterations[J]. IEEE Transactions on Computers, 1993, 23(10): 993 – 1001.

[5] Hu Y H. The quantization effects of the CORDIC algorithm [J]. IEEE Transactions on Signal Processing, 1992, 40(4): 705 – 707.

作者简介:

孙　学(1978—),男,重庆合川人,2004 年于重庆大学获硕士学位,现为工程师,主要研究方向为交换式总线、分布式计算和阵列计算技术。

SUN Xue was born in Hechuan, Chongqing, in 1978. He received the M.S. degree from Chongqing University in 2004. He is now an engineer. His research concerns switch fabric, distributed computing and array computing.

Email: sun8xue@163.com