文章编号:1671 - 6833(2008)02 - 0084 - 04

正弦和余弦函数的一种混合式 CORDIC 算法实现

盛利元, 孔德元, 简远鸣, 马剑钊

(中南大学 物理科学与计算技术学院, 湖南 长沙 410083)

摘 要: 在传统的 CORDIC 算法的基础上提出一种改进算法,然后给出实现正弦、余弦函数的运算的混合算法. 该算法在不影响数据精度要求的情况下,不仅可以减少迭代次数和所需的 ROM 存储空间,还有利于结构的流水线设计,减少系统的时钟周期,提高速度. 最后在 Altera 公司的 Cyclone 系列芯片 EP1C3T100C6 上予以实现. 仿真结果证明:此算法比传统算法具有高运算速度与低资源的优势,最高工作频率可达到 184.77 MHz,比传统算法提高了 11.84%,在结构上较传统的 CORDIC 算法节省近 17.35% 硬件资源.

关键词: CORDIC 算法;混合算法;FPGA

中图分类号: TN 431.2;TN 918.2 **文献标识码**: A

0 引言

在现代数字信号处理中,经常需要快速和精确地进行一些特殊三角函数值计算. 早期的方法是采用查表、多项式展开或近似的方法,这些方法在速度、精度、资源等方面不能很好地兼顾[1]. 坐标旋转算法(Coordinate Rotation Digital Computer, CORDIC)可以兼顾这些方面,它能够将多种难以用硬件电路直接实现的复杂运算分解为统一的加减、移位操作,极大地降低了硬件设计的复杂性. 近几年,随着现代微电子技术的快速发展,用硬件实现以往用软件进行的计算得到很多专家的重视. 因此对该算法进行研究具有重要意义.

在传统的 CORDIC 算法中,对于运算位宽为 n 位的 CORDIC 算法,其劣势主要表现为 2 个方面:一是随着 n 的增加,ROM 的容量会成指数增长[2-4],占用更多的硬件资源;二是 CORDIC 算法至少需要 n 级流水线,并且每级迭代需要查 ROM 表,不利于系统的流水线设计,容易形成速度"瓶颈",从而降低了系统的性能.

针对这些问题,笔者提出了一种改进的 CORDIC 算法,然后利用传统的 CORDIC 算法以及与这种改进算法相结合的混合算法实现对正弦、余弦函数的运算. 该混合算法不仅可以减少迭代次数和 ROM 存储单元数量,而且还可以缩短演算所执行的时间、降低硬件成本、缩小晶片面积.

1 传统的 CORDIC 算法原理

CORDIC 算法最初是 Volder[5-6] 于 1959 年提出,其基本思想是通过一系列固定的、与运算基数相关的角度不断偏转从而逼近所需旋转的角度. 下面就简要地介绍一下 CORDIC 算法的基本思想. 如图 1 所示,将向量 (x_i, y_i) 旋转 θ 角,得到一个新的向量 (x_j, y_j),这个关系用矩阵表示如下:

$$\begin{bmatrix} x_j \\ y_j \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \cos\theta \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (1)$$

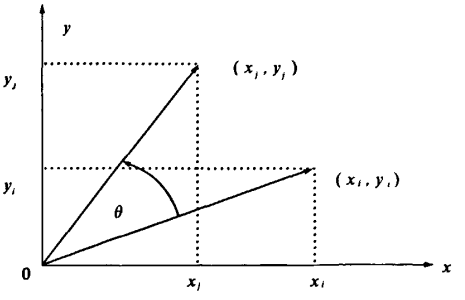

图 1 CORDIC 算法原理图

Fig. 1 Diagram of CORDIC algorithm principle

如果假设旋转角度 θ 是由连续 n 个微小的角

收稿日期:2008 - 03 - 19;修订日期:2008 - 05 - 04

基金项目:国家自然科学基金资助项目(60672041)

作者简介:盛利元(1956 -),男,湖南益阳人,中南大学教授,主要研究方向为混沌理论和数字信号处理.

度叠加而成的，即 $\theta=\sum_{k=0}^{n-1}\delta_k\theta_k$. 那么根据式(1)得出每一步的叠加操作需要按照下式进行

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

向量(x_i,y_i)经过 n 步旋转之后到达向量(x_j,y_j)有如下表示：

$$\begin{bmatrix}x_j\\ y_j\end{bmatrix}=\prod_{k=0}^{n-1}\cos\theta_k\begin{bmatrix}1 & -\delta_k2^{-k}\\ \delta_k2^{-k} & 1\end{bmatrix}\begin{bmatrix}x_i\\ y_i\end{bmatrix}$$

$$=K*\prod_{k=0}^{n-1}\begin{bmatrix}1 & -\delta_k2^{-k}\\ \delta_k2^{-k} & 1\end{bmatrix}\begin{bmatrix}x_i\\ y_i\end{bmatrix} \quad (3)$$

为了方便计算，我们选取 $\theta_k=\arctan(2^{-k})$，同时引入一个新的变量 z_n，z_n 代表剩下的未旋转的角度.

$$z_n=\theta-\sum_{k=0}^{n-1}\delta_k\theta_k \quad (4)$$

其中

$$\delta_k=\begin{cases}-1, & if\ z_n<0\\ 1, & if\ z_n>0\end{cases} \quad (5)$$

(3)式中 K 为模校正因子，取极限有

$$K=\prod_{k=0}^{\infty}\cos\theta_k=\prod_{k=0}^{\infty}\frac{1}{\sqrt{1+2^{-2k}}}\approx 0.607\ 253 \quad (6)$$

如果我们在设计的系统中提前计算 K 值，可得传统 CORDIC 算法的圆周旋转模式表达式[7]

$$\begin{cases}x_{n+1}=x_n-\delta_ny_n2^{-n}\\ y_{n+1}=y_n+\delta_nx_n2^{-n}\\ z_{n+1}=z_n-\delta_n\arctan(2^{-n})\end{cases} \quad (7)$$

$\arctan(2^{-n})$的值可以预先求出，并保存在常数 ROM 中. 选择特定的初值，$x_0=\frac{1}{K}$，$y_0=0$，$z_0=\theta$，n 次迭代后，$x_n\to\cos\theta$，$y_n\to\sin\theta$. 因此，可以求出正弦函数和余弦函数的值. 由于 θ 角的最大值 θ_{max} 满足[8-9]

$$\theta_{max}=\sum_{i=0}^{\infty}\arctan(2^{-i})\approx 99.9^\circ \quad (8)$$

为了扩大输入角度的范围，需要对输入角度进行 90°取模的预处理[10]，为了方便运算，笔者假设 θ 角满足 $0\leqslant\theta\leqslant\frac{\pi}{2}$.

2　混合式的 CORDIC 算法

该算法的整个迭代过程可分为两部分进行，前部分采用传统的 CORDIC 公式迭代，后部分采用改进 CORDIC 公式迭代. 笔者先从每次旋转的角度 $\theta_i=arctan(2^{-i})$，$(i=0,1,2,\cdots,n-1)$为研究出发点.

利用泰勒级数将 $\theta_i=\arctan(2^{-i})$展开为

$$\theta_i=\arctan(2^{-i})=2^{-i}-\frac{1}{3}\cdot2^{-3i}+\frac{1}{5}\cdot2^{-5i}-K$$

$$=2^{-i-1}+2^{-i-2}+K \quad (9)$$

因此 θ_i 为 $\theta_i=0.00\cdots0.11\theta_i^{i+3}\theta_i^{i+4}\cdots$，且 $\theta_i^{i+j}\in\{0,1\}$，$(j=3,4,5,\cdots)$. 从式(9)中我们可以得出，随着值的 θ_i 减小，θ_i 又可以表示为 $\theta_i=0.00\cdots011\cdots11$ 的形式. 经过初始几次迭代后，2^{-i} 与 $\arctan(2^{-i})$之差迅速减小. 假设当迭代到第 m 级时可以用 2^{-m}代替 $\arctan(2^{-m})$，此时迭代方程就可以不需要查 ROM 表，加速了系统迭代的速度. 关键是如何确定 m 的值，使得用 2^{-m}代替 $\arctan(2^{-m})$来进行运算时产生的误差可以忽略不计. 这时 m 值应该满足

$$\sum_{i=m}^{n-1}(2^{-i}-\arctan(2^{-i}))\leqslant 2^{-n} \quad (10)$$

将式(9)代入上式，可得

$$m=\left\lceil 1-\frac{1}{3}\log_2^{(21.2^{-n}+8.2^{-3n})}\right\rceil \quad (11)$$

式中$\lceil*\rceil$表示 $*$ 的最大整数. 取 $n=16$，可以算出 $m=5$. 因为

$$\cos\theta_i=\frac{1}{\sqrt{1+2^{-2i}}}=1-2^{-2i-1}+3\cdot2^{-4i-3}+\Lambda$$

$$=1-2^{-2i-1}+O(2^{-31})\ \text{当}\ i\geqslant7 \quad (12)$$

所以，当 $n\geqslant7$ 时，公式(2)可改写为：

$$\begin{bmatrix}x_{n+1}\\ y_{n+1}\end{bmatrix}=(1-2^{-2n-1})\begin{bmatrix}1 & -\delta_n\cdot2^{-n}\\ \delta_n\cdot2^{-n} & 1\end{bmatrix}\begin{bmatrix}x_n\\ y_n\end{bmatrix} \quad (13)$$

对于运算位宽为 16 位的 CORDIC 算法迭代，当$(n\geqslant7)$时，有 $1-2^{-2n-1}\approx1$ 成立，综合式(7)和(13)两式有

$$\begin{cases}x_{n+2}=x_{n+1}-\delta_{n+1}2^{-n-1}y_{n+1}\\ \quad=x_n-(\delta_n2^{-n}+\delta_{n+1}2^{-n-1})y_n\\ y_{n+2}=y_{n+1}+\delta_{n+1}2^{-n-1}x_{n+1}\\ \quad=y_n+(\delta_n2^{-n}+\delta_{n+1}2^{-n-1})x_n\\ z_{n+2}=z_{n+1}-\delta_{n+1}\arctan(2^{-n-1})\\ \quad=z_n-\delta_n\arctan(2^{-n})-\delta_{n+1}\arctan(2^{-n-1})\end{cases} \quad (14)$$

成立. 从(14)式可以看出，当我们选择适当的(δ_n,δ_{n+1})时，可以在一次迭代运算中处理未旋转角 z_n 的相邻的 2 个位元，加速未旋转角度趋近于，进一步减少流水线的级数. 对于初始的 θ 角，

当我们进行了 $n(n\geq 7)$ 次旋转后,一定有 $|z_n|<1$ 成立,所以 z_n 的二进制表示形式为 $z_n = z_n^0 0.z_n^1 z_n^2 z_n^3 \Lambda$,其中 z_n^0 为符号位. 如果 z_n 为正数,则 $z_n^0=0$;如果 z_n 为负数,则 $z_n^0=1$. 当 $z_n>0$(即 $z_n^0=0$)时,我们分以下4种情况来确定 (δ_n,δ_{n+1}) 的值.

(1)如果 $z_n^{n+1}z_n^{n+2}=11$,若取 $\delta_n=1,\delta_{n+1}=0$,则由(15)式可得 $z_{n+2}=z_n-\theta_n$,即

$$\begin{array}{l} z_n = 00.0\cdots 11 z_n^{n+3} z_n^{n+4}\cdots \\ -\theta_n = 0.00\cdots 11\ 11\cdots \\ \hline z_{n+2} = z_{n+2}^0 0.0\cdots 00 z_{n+2}^{n+3} z_{n+2}^{n+4}\cdots \end{array}$$

(2)如果 $z_n^{n+1}z_n^{n+2}=00$,若取 $\delta_n=0,\delta_{n+1}=0$,则由(15)式可得 $z_{n+2}=z_n-0$,即

$$\begin{array}{l} z_n = 00.0\cdots 00 z_n^{n+3} z_n^{n+4}\cdots \\ -\theta_n = 0.00\cdots 0000\cdots \\ \hline z_{n+2} = z_{n+2}^0 0.0\cdots 00 z_{n+2}^{n+3} z_{n+2}^{n+4}\cdots \end{array}$$

(3)如果 $z_n^{n+1}z_n^{n+2}=01$,若取 $\delta_n=0,\delta_{n+1}=1$,则由(15)式可得 $z_{n+2}=z_n-\theta_{n+1}$,即

$$\begin{array}{l} z_n = 00.0\cdots 01 z_n^{n+3}\cdots \\ -\theta_{n+1} = 0.00\cdots 011\cdots \\ \hline z_{n+2} = z_{n+2}^0 0.0\cdots 00 z_{n+2}^{n+3}\cdots \end{array}$$

(4)如果 $z_n^{n+1}z_n^{n+2}=10$,若取 $\delta_n=0,\delta_{n+1}=1$,则由(15)式可得 $z_{n+2}=z_n-\theta_{n+1}$,即

$$\begin{array}{l} z_n = 00.0\cdots 10 z_n^{n+3} z_n^{n+4} z_n^{n+5}\cdots \\ -\theta_{n+1} = 0.00\cdots 01111\cdots \\ \hline z_{n+2} = z_{n+2}^0 0.0\cdots 0 z_{n+2}^{n+2} z_{n+2}^{n+3} z_{n+2}^{n+4} z_{n+2}^{n+5}\cdots \end{array}$$

从上述4种情况可以看出,只有在第4种情况下,$z_{n+2}^{n+1}z_{n+2}^{n+2}$ 的值不一定为00,有可能为01. 当 $z_{n+2}^{n+1}z_{n+2}^{n+2}$ 的值为01时,为了达到同时处理未旋转角 z_n 的相邻两位位元,需要对 z_n 的值进行修正. 采用的方法是将 z_n 的值减去 2^{-15},虽然此时会产生较小的误差,但是相对于16位的CORDIC算法迭代系统的精度要求可以忽略不计.

对于 $z_n<0(z_n^0=1)$ 的情况,我们也可以利用上述的推导方法来确定 (δ_n,δ_{n+1}) 的值,(δ_n,δ_{n+1}) 的取值规则见表.

表1 (δ_n,δ_{n+1}) 取值表

Tab.1 Dereferencing table of (δ_n,δ_{n+1})

(δ_n,δ_{n+1})	$(z_n^0 z_n^{n+1} z_n^{n+2})$	(δ_n,δ_{n+1})	$(z_n^0 z_n^{n+1} z_n^{n+2})$
(1,0)	(011)	(-1,0)	(100)
(0,1)	(001)或(010)	(0,-1)	(101)或(110)
(0,0)	(000)或(111)		

因此,可得出改进的CORDIC公式为:

$$\begin{cases} x_{n+2} = x_n - (\delta_n 2^{-n} + \delta_{n+1} 2^{-n-1}) y_n \\ y_{n+2} = y_n - (\delta_n 2^{-n} + \delta_{n+1} 2^{-n-1}) x_n \\ z_{n+2} = z_n - \delta_n \cdot 2^{-n} - \delta_{n+1} 2^{-n-1} \end{cases} \quad (15)$$

上式中的 (δ_n,δ_{n+1}) 取值满足表1的规则. 根据改进的CORDIC式(15),结合传统的CORDIC式(7),我们就可以采用二者相结合的混合CORDIC算法对正弦、余弦函数进行运算. 由于 $(\delta_n,\delta_{n+1})\neq(1,1)$,则(15)式中 x_n 和 y_n 的系数实际上只有一项. 由前面的分析知,首先用公式(7)进行八次迭代,这时模校正因子应修正为 $K=\prod_{n=0}^{7}\frac{1}{\sqrt{1+2^{-2n}}}\approx 0.607.259$. 后面的8级直接用式(15)迭代,只需迭代次,并且模校正因子近似为,不必进行修正. 因此整个过程只需迭代次,减少了级流水线,8个ROM存储单元,降低了CORDIC单元的硬件消耗,并且后面的乘法操作可用并行加法完成[11],缩短了系统的运算时间. 因此,在不影响系统性能的情况下,这种混混合式的CORDIC算法节省了硬件资源和时间. 下面通过FPGA的仿真我们可以清楚地看出它具有明显的优点.

3 FPGA算法实例

该系统采用Altera公司的Cyclone系列芯片EP1C3T100C6,利用Verilog语言编程,在Quartus II开发平台上实现. 两种算法仿真时的时钟周期均为10 ns,算法的位宽都16位,所取的角度分别为0°,30°,45°,60°,90°. 2种方法的部分仿真结果如表2所示. 由于CORDIC算法本身是一种近似计算,所以2种结果与标准值之间都存在误差. 但误差范围都在同一数量级上,应属算法允许范围之内. 传统的CORDIC算法在15个时钟周期起始延迟之后,在每次新的循环之后就会生成一个新的输出值,而混合式的算法只需要11个时钟周期,很显然提高了运算的速度. 最后通过Synplify7.6工具综合,综合结果显示,传统方法门数为3 891,混合方法的门数为3 216,比传统方法降低了17.35%;最高工作频率由原来的165.21 MHz提高到184.77 MHz,提高了11.84%.

4 结论

笔者以传统的CORDIC算法为基础,设计了一种改进的算法,结合两种方法成功地实现了正弦和余弦函数的计算,并且在不影响系统性能的情

表 2　2 种方法的数据比较

Tab. 2　The contrast about data of this two method

角度		标准值	传统方法		误差	本文混合方法		误差
0°	sin	0.000 00	0x01CC	0.014 03	1.403×10^{-2}	0x0186	0.011 90	1.19×10^{-2}
	cos	1.000 00	0x8000	1.000 00	0.0	0x8000	1.000 00	0.0
45°	sin	0.707 11	0x5A82	0.707 09	2.0×10^{-5}	0x5A7F	0.707 02	-9.0×10^{-5}
	cos	0.707 11	0x5A83	0.707 12	1.0×10^{-5}	0x5A84	0.707 15	4.0×10^{-5}
60°	sin	0.866 03	0x6EDC	0.866 09	6.0×10^{-5}	0x6EDB	0.866 06	3.0×10^{-5}
	cos	0.500 00	0x4000	0.500 00	0.0	0x4004	0.500 12	1.2×10^{-4}

况下,减少了硬件资源的消耗,提高运算速度. 该混合式的 CORDIC 算法为我们进行位宽为 16 位和 32 位的 CORDIC 运算提供了重要思路,而且随着运算字长位宽的增大,其速度及资源消耗方面的优势更加明显. 因此混合式的 CORDIC 算法有着很好的发展前景,值得我们进一步研究和应用.

参考文献:

[1] 李　滔,韩月秋. 基于流水线 CORDIC 算法的三角函数发生器[J]. 系统工程与电子技术, 2000, 2(4):85 - 87.

[2] 周　柱,张　炜. 基于 CORDIC 的优化的直接数字频率合成器[J]. 电子工程师, 2005, 31(10): 34 - 37.

[3] TIMMERMAN D,HAHN H,HOSTICKA B J. Low latency time CORDIC algorithms[J]. IEEE Trans. on Computer,1992, 41(8):1010 - 1014.

[4] TAKAGI N, ASADA T, YAJIMA S. Redundant CORDIC methods with a constant scale factor for sine and cosine computation[J]. IEEE Trans. On Computers, 1991, 40:989 - 995.

[5] VOLDER J E. The CORDIC trigonometric computing technique [J]. IEEE Trans. on Electronic Computering,1959, EC - 8:330 - 334.

[6] WALTHER J S. A unified algorithm for elementary functions[A]. Proc. AFIPS Spring Joint Computer Conference, 1971, 379 - 385.

[7] PIRSH P. Architectures for digital signal processing [M]. John Wiley & Sons, 1998.

[8] WANG S, PIURI V, WARTZLANDER E. Hybrid CORDIC algorithms [J]. IEEE Trans on Computers, 1997, 46(11):1202 - 1207.

[9] CHRISTOPHE M. Computing functions arccros arcsin using cordic [J]. IEEE Trans on Computer, 1993, (142): 235 - 239.

[10] WU A Y, WU C S. A unified view for vector rotational CORDIC algorithms and architecture based on angle quantization approach [J]. IEEE Transaction and Systems - 1, 2002, 49(10):1442 - 1445.

[11] 鞠建波,别　庆,杜爱国. 基于改进的 CORDIC 算法 QDD 的 FPGA 实现及精度分析[J]. 电视技术, 2007,(2): 112 - 116.

An Implementation of Sine and Cosine Hybrid CORDIC Algorithm

SHENG Li - yuan, KONG De - yuan, JIAN Yuan - ming, MA Jian - zhao

(School of Physics Science and Technology, Central South University , Changsha 410083,China)

Abstract: The paper proposes an improved algorithm and a hybrid algorithm to realize sine and cosine computation based on traditional CORDIC algorithm. This method on the precondition of uninfluencing the data Accuracy, can not only reduce iterative times and ROM store space but advance the structure's pipeline design to reduce structure's clock cycle in higher speed. Its maximal operation frequency can reach 184.77 MHZ which is increased by 11.84%, and 17.35% hardware source can be saved compared with traditional CORDIC algorithm.

Key words: CORDIC algorithm; hybrid algorithm; FPGA