

一种 80 位扩展双精度浮点三角函数运算单元的设计

张 佳, 唐 威, 盛廷义, 赵文琦, 艾 刁

(西安微电子技术研究所, 陕西 西安 710054)

摘 要: 三角函数求值这一运算计算过程复杂, 硬件较难实现. 针对这一问题, 通过改进 CORDIC 算法, 实现了兼容 SPARC 处理器和 INTEL 处理器浮点标准的 80 位高精度浮点三角函数的计算. 在算法设计中, 将函数的计算范围扩展至 $-\pi \sim +\pi$, 并且实现了迭代次数的可配置. 最后验证了算法的正确性与完整性, 分析了运算过程中迭代次数与精度的关系. 结果表明, 运算的精度提高到 10^{-14} .

关键词: CORDIC 算法; 高精度; 三角函数; 浮点

中图分类号: TN402

文献标识码: A

文章编号: 1000-7180(2014)04-0023-04

Design of a 80 Bits Extended Double Precision Floating-Point Trigonometric Computing Unit

ZHANG Jia, TANG Wei, SHENG Ting-yi, ZHAO Wen-qi, AI Diao

(Xi'an Microelectronics Technology Institute, Xi'an 710054, China)

Abstract: The trigonometric algorithm is hard to achieve. By improving the CORDIC algorithm achieved a 80-precision floating-point trigonometric calculation which compatibled with the INTEL processor and SPARC processor. In the design, the calculation range has been extended to $-\pi \sim +\pi$, and the number of the iterations can be configured. Finally, the correctness and completeness of the algorithm are verified, and the relationship between the number of iterations and the accuracy was analyzed. The results showed that the precision was raised up to 10^{-14} , the simplicity and speed of the algorithm were also great advantages.

Key words: CORDIC algorithm; high-precision; trigonometric functions; floating-point

1 引言

三角函数的求值作为计算科学中一种重要运算, 涉及到的科学领域非常广泛. 目前三角函数实现方法主要有查找表法、多项式拟合法等. 查找表法只能实现单精度浮点运算, 多项式拟合法收敛较慢, 硬件实现也比较复杂^[1-3]. 而 CORDIC 算法在实现过程中只有移位和加减运算, 很好地避免了以上问题. 此外, 目前通过应用 CORDIC 算法实现三角函数计算的算法中, 设计精度都比较低, 在很多领域难以满足要求.

本文通过对 CORDIC 算法的改进, 使得迭代次数可配置, 从而实现了可根据精度要求配置迭代次数这一目标, 完成了 80 位扩展双精度浮点三角函数

的计算. 改进的算法兼顾了速度、精度、效率等因素, 尤其在高精度方面有很大的优势.

2 CORDIC 算法简介^[4-5]

假设一个直角坐标系内有一个向量 $a(X_a, Y_a)$, 逆时针旋转角度 θ 后得到另一个向量 $b(X_b, Y_b)$, 如图 1 所示.

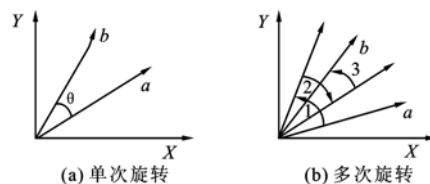


图 1 算法示意图

则向量 b 与向量 a 的关系为

$$\begin{cases} X_b = \cos\theta(X_a - Y_b \tan\theta) \\ Y_b = \cos\theta(Y_a - X_b \tan\theta) \end{cases} \quad (1)$$

令 $\tan\theta = \pm 2^{-i}$, 式(1) 中乘以正切项可以通过简单的移位操作实现. 而任意角度的变换都可以通过一系列的微旋转来完成. 每次迭代相当于旋转角度 $\theta_i = \arctan(2^{-i})$, 则 $\cos\theta_i$ 为常数, Z_i 的符号决定旋转的方向(在向量模式中, 由 Y_i 的符号确定旋转方向), 用 σ_i 表示, 迭代的角 θ_i 总和即为所求的角度. 则迭代的公式可表示为

$$\begin{cases} X_{i+1} = K_i(X_i - Y_i \sigma_i 2^{-i}) \\ Y_{i+1} = K_i(Y_i + X_i \sigma_i 2^{-i}) \\ Z_{i+1} = Z_i + \sigma_i \arctan(2^{-i}) \end{cases} \quad (2)$$

其中, $\sigma_i = \begin{cases} +1, Z_i < 0 \\ -1, Z_i > 0 \end{cases}$, $K_i = \cos(\arctan(2^{-i}))$, $Z_0 = 0$.

随着旋转次数的增多, K_i 收敛于:

$$K = \prod_{i=1}^{\infty} \cos(\arctan(2^{-i})) \approx 0.607\ 253. \quad (3)$$

在旋转模式下, 若对一组初始化值^[6]: $X_a = K \approx 0.607\ 259\ 350\ 088\ 881\ 558\ 845\ 050\ 570\ 496$, $Y_a = 0$, $Z_a = \theta$, 那么, 当 Z 趋近于 0 时, 得到的迭代结果为: $X_n = \cos\theta$, $Y_n = \sin\theta$, $Z_n = 0$.

在向量模式下, 若对一组初始化值: $X_a = 1$, $Y_a = \theta$, $Z_a = 0$, 那么, 当 Y 趋近于 0 时, 得到的迭代结果为: $X_n = \frac{1}{K} \sqrt{\theta^2 + 1}$, $Y_n = 0$, $Z_n = \arctan(\theta)$.

3 运算单元的设计

在基于 CORDIC 的算法设计的运算单元中, 完成设计还需要考虑一些具体的因素, 并对算法做出相应的改进. 这些改进包括以下三个方面:

一方面, SPARC 处理器和 INTEL 处理器对 80 位浮点数的表示有所不同, 在本文中, 程序是在 SPARC 处理器下运行的. 为了使算法有广泛的适用性, 对 INTEL 处理器下程序的运行进行了兼容性设计. IEEE-754 标准^[4]规定, 每个浮点数由 3 部分构成: 符号位 S、指数部分 E 和尾数部分 M. 在 SPARC 环境下和在 INTEL 环境下的 80 位浮点数可用图 2 表示^[7].

由图 2 可知, SPARC 格式的 80 位扩展双精度所表示的数为

$$n = (-1)^S \times (1.M) \times (2^{R-16\ 383}).$$

而 INTEL 格式的 80 位扩展双精度所表示的数为



图 2 80 位扩展双精度的 IEEE-754 表示

$$n = (-1)^S \times (j.M) \times (2^{R-16\ 383}).$$

另一方面, 由于 CORDIC 算法采用迭代思想, 算法所能覆盖的角度范围有限, 而为了满足更大范围的计算需求, 需要对计算范围进行扩展. 算法本身所能计算的角度范围 θ 与迭代的次数 n 有关, 设每次迭代的角为 θ_i , 则角度范围 $\theta = \sum_{i=1}^n \theta_i$, 当 $i \geq 13$ 时, $\theta \approx 99.88^\circ$, 即最大覆盖角度范围为 $-99.88^\circ \sim +99.88^\circ$. 为了将计算范围扩展到整个正弦或余弦周期 ($-\pi \sim +\pi$), 本文运用三角函数的诱导公式进行处理.

此外, 还解决了小角度的三角函数计算问题. 当输入一个很小角度 θ 时, 规定了如果 $\theta < T$, 则它的三角函数值为

$$\begin{cases} \sin\theta = \theta \\ \arctan\theta = \theta \\ \cos\theta = 1 - \theta \end{cases} \quad (4)$$

由于算法所能计算的角度不可能无限小, 它有一个最小分辨角度 T , 当所要计算的角度小于最小分辨角度 T 时, 就会产生错误. 而上文的设计就避免了这种错误的发生, 从而增加了算法的完整性, 并且提高了运算速度. 这里需要说明, T 并不是一个一成不变的数, 它随着迭代次数的变化而变化, 规定 $T = 2^{-n}$, 其中, n 为配置的迭代次数.

本文实现 80 位高精度浮点三角函数的流程图如图 3 所示, 其中, 数据选择操作的作用是选择初始输入的一组特殊值, 并把它保存到寄存器中. 所选择的特殊值根据所求角的三角函数种类确定, 当计算正余弦时, 三个寄存器分别选择 K 、0、Data; 当计算反正切时, 三个寄存器分别选择 1、Data、0. 然后分别对 X 、 Y 寄存器中的数据进行移位, 如公式(2)所示, 使 X_i 、 Y_i 分别除以 2, 表现在算法中即为左移一位, 完成移位后, 再分别与 Y_i 、 X_i 进行加减, 加减符号位由 Z (正余弦)或者 Y (反正切)的符号 σ_i 和公式本身的符号共同确定. 加减计算完成后, 把结果保存在结果寄存器中, 这样就完成了一次迭代. 算法本身是一个迭代与循环的过程, 所以, 每完成一次计算,

把新的计算结果再存入 X、Y、Z 寄存器,开始下一次迭代.直到完成多次迭代循环,计算结束.迭代次数可根据需要的精度进行配置.迭代完成后输出结果寄存器中的数值.

通过上述的算法描述,可以看到,算法的实现中只有移位和加减运算,过程简单,每次迭代只需要 3 个时钟周期,速度快,特别适合硬件实现.

4 结果与讨论

在算法的设计中,实现了迭代次数的可配置,而上文中也提到迭代次数与计算精度有关.表 1 列出了迭代次数与计算精度的关系,取 0.52 弧度时的

sin 值作为参考.

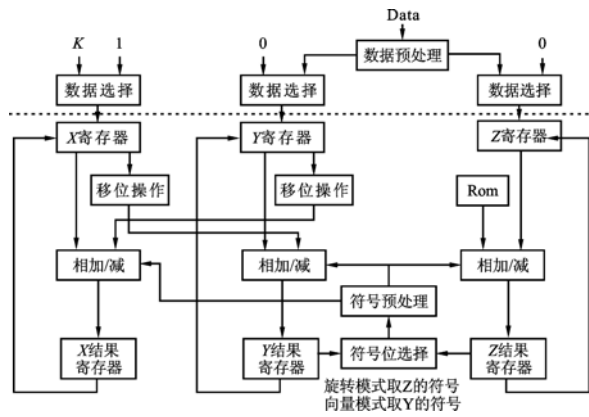


图 3 算法流程图

表 1 精度与迭代次数关系

迭代次数	参考值	实际值	精度范围
10 次	3FFDFCCE2598488556B0	3FFDFB6EB16BDDE508FC	10^{-2}
30 次		3FFDFCCE25BCCD0015A5	10^{-8}
40 次		3FFDFCCE259B490C127A	10^{-11}
53 次		3FFDFCCE259B4885C081	10^{-14}
64 次		3FFDFCCE259B48859D4C	10^{-14}

由表 1 可知,当迭代次数为 10 次时,精度只能达到 10^{-2} ,而当迭代次数为 30 次时,精度很快收敛到了 10^{-8} .随着迭代次数增多,计算精度也在提高.当迭代次数大于 53 次后,精度的改善将变得不再显著.并且迭代次数越多,需要的计算周期就越长,相应地也会

占用更多资源,所以在一定精度要求下,用户可以适当地配置迭代次数以达到精度与效率的折衷.

用 Questasim 软件对本设计进行验证,分别随机选取弧度为 2^{-65} 、 $-\pi/4$ 、 $\pi/6$ 、 $\pi/2$ 、 $3\pi/5$ 等 5 个值迭代 64 次,计算结果如表 2 所示.

表 2 仿真结果统计

弧度	函数名	理论值	计算值	误差
2^{-65}	sin	3FBE0000000000000000	3FBE0000000000000000	$<10^{-15}$
	cos	3FFEFFFFFFFFFFFFFFFFFFFF	3FFBFFFFFFFFFFFFFFFFFFFF	$<10^{-15}$
	atan	3FFBFFFFFFFFFFFFFFFFFFFF	3FBE0000000000000000	$<10^{-15}$
$-\pi/4$	sin	BFFE6A09E667F3BCC908	BFFE6A09E667F3BCF75E	10^{-14}
	cos	3FFE6A09E667F3BCC908	3FFE6A09E667F3BCFF45	10^{-14}
	atan	BFFE54E04C05D06A06CE	BFFE54E04C05D06A0A0C	10^{-15}
$\pi/6$	sin	3FFE0000000000000000	3FFE0000000000001767	$<10^{-15}$
	cos	3FFE6B67AE8584CAA73B	3FFE6B67AE8584CAEBD7	10^{-14}
	atan	3FFDEDEC9C192BFAACDE	3FFDEDEC9C192BFAB57B	$<10^{-15}$
$\pi/2$	sin	3FFF0000000000000000	3FFF0000000000002390	$<10^{-15}$
	cos	00000000000000000000	BFC90B3AAD329C9866E0	$<10^{-15}$
	atan	3FFE00FE987ED02FF379	3FFE00FE987ED02FF27F	$<10^{-15}$
$3\pi/5$	sin	3FFEE6F0E134454FF5E6	3FFEE6F0E13445503AC9	10^{-14}
	cos	BFFD3C6EF372FE94F82B	BFFD3C6EF372FE951C53	10^{-14}
	atan	3FFF1541C1BE7C1702BB	3FFF1541C1BE7C170005	$<10^{-15}$

为了检验算法的计算精度,同时验证算法的完整性与正确性,所选取的检验值都有一定意义.其

中, 2^{-65} 代表了小角度数值, $-\pi/4$ 代表了 $-\pi/2 \sim +\pi/2$ 内的负数, $\pi/6$ 代表了 $-\pi/2 \sim +\pi/2$ 内的正数, $\pi/2$ 代表了一个特殊值, $3\pi/5$ 代表了 $-\pi \sim +\pi$ 内的正数. 通过表 2 可以看出, 算法很好地解决了小角度数值与大角度 ($-\pi \sim +\pi$ 内) 数值的三角函数计算问题, 迭代精度最低为 10^{-14} . 而采用多项式拟合法计算复杂, 也很难达到本文的计算精度.

图 4 表示了采用多项式拟合法^[8]得到的精度与拟合项数的关系, 其中, 横轴表示需要拟合的项数, 纵轴表示所能达到的精度与拟合项数的关系. 通过图 4 可看出, 多项式拟合法在拟合项数大于 5 以后, 精度几乎不再变化, 提高精度将变得很困难, 并且需要多步的浮点数乘法. 而改进后的 CORDIC 算法不仅精度非常高, 而且只需要移位和加减运算, 更适合硬件实现.

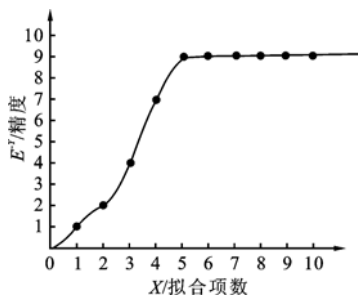


图 4 多项式拟合法精度与拟合项数关系

5 结束语

本文通过改进的 CORDIC 算法, 实现了迭代次数的软件可配置. 同时, INTEL 处理器与 SPARC 处理器对 80 位浮点数表示的差别问题也得到了处

理. 本文还拓展了计算范围, 同时对小角度计算进行了处理, 使运算更加完善. 最终, 实现了高速、高精度的三角函数运算, 配置灵活, 精度高, 适合在嵌入式计算机子系统中广泛应用.

参考文献:

- [1] 孔德元. 针对正弦余弦计算的 CORDIC 算法优化及其 FPGA 实现[D]. 长沙: 中南大学, 2008.
- [2] 陈石平, 李全, 付佃华, 等. 32 位浮点正余弦函数的 FPGA 实现[J]. 微计算机信息, 2008, 24(5): 3-5.
- [3] 潘宏亮. 浮点指数类超越函数的运算算法研究与硬件实现[D]. 西安: 西北工业大学, 2006.
- [4] Oskar Mencer. Application of reconfigurable CORDIC architectures[J]. Journal of VLSI Signal Processing Systems, 2000, 24: 211-221.
- [5] Zhou Zhiheng, Wu Dongcheng. An algorithm for computing the value of arbitrary angle trigonometric function Transportation[C]//2011 International Conference on Digital Object Identifier, Guangzhou: South China University of Technology, 2011.
- [6] IEEE organise. IEEE Std 754-1985 IEEE Standard for Binary Floating-Point Arithmetic [S]. New York: IEEE, 1985.
- [7] 朱亚超. 基于 IEEE-754 的浮点数存储格式分析研究[J]. 计算机与信息技术, 2006(9): 50-52.
- [8] 司少玲, 关永. 三角函数曲线数据拟合最佳次数的确定[J]. 计算机工程与设计, 2006, 27(24): 4660-4662.

作者简介:

张 佳 男, (1986—), 硕士研究生. 研究方向为 IP 设计.

(上接第 22 页)

- [6] 张锐娟, 张建奇, 杨翠. 基于 SURF 的图像配准方法研究[J]. 红外与激光工程, 2009, 38(1): 160-165.
- [7] 贺柏根, 朱明. 改进的抗全仿射尺度不变特征变换图像匹配方法[J]. 光学精密工程, 2012, 41(3): 811-817.
- [8] 李小昌, 朱丹. 采用尺度不变特征和区域选择的图像配准方法[J]. 红外与激光工程, 2012, 41(2): 537-542.

作者简介:

郭晓冉 男, (1985—), 博士研究生. 研究方向为电子稳像、目标检测、识别与跟踪技术.

崔少辉 男, (1965—), 博士生导师, 教授. 研究方向为目标检测、识别与跟踪技术以及弹载仪器与微系统.

方 丹 男, (1979—), 博士, 讲师. 研究方向为目标检测、识别与跟踪技术.