

32 位定浮点数正余弦函数 FPGA 实现方法

吴庆达, 何书专, 潘红兵, 沙 金, 李 伟

(南京大学 微电子设计研究所, 江苏 南京 210093)

摘 要: 本文首先介绍了查表算法和 Cordic 算法原理, 在这两种算法基础上, 用 Verilog HDL 语言对 32 位定点数的正余弦函数进行了编程设计, 结合仿真综合结果, 对这两种方法从运算精度, 运算速度和占用硬件资源几方面进行了分析. 进而采用不经过浮点定点转换, 直接在 Cordic 算法改进的基础上实现 32 位浮点数的正余弦函数 FPGA 设计. 最后, 对这三种实现方法进行了综合评价.

关键词: 查表法, Cordic 算法, 三角函数

中图分类号: TP39

文献标识码: A

文章编号: 1000-7180(2012)01-0113-04

Implementations of 32-bits Fixed and Floating Point Trigonometric Functions with FPGA

WU Qing-da, HE Shu-zhuan, PAN Hong-bing, SHA Jin, LI Wei

(Institute of VLSI Design, Nanjing University, Nanjing 210093, China)

Abstract: Firstly, this paper introduces the Lookup-table algorithm and cordic algorithm. Based on the two algorithms, we program and design 32-bit fixed-point trigonometric functions with Verilog-HDL. Combined with simulation, we evaluate these two methods by comparing operation precision, operation speed and hardware resources occupied. Then, we design the 32-bit floating-point trigonometric functions directly based on improved cordic algorithm without floating-fixed exchange. Finally, we evaluate these three methods.

Key words: lookup-table algorithm; Cordic algorithm; trigonometric functions

1 引言

数字信号处理中经常需要用到三角函数的运算. 在数字设计上实现三角函数, 通常有查表法, Cordic 算法和级数展开法等. 级数展开法由于其展开式的运算复杂性, 不易采用 FPGA 实现, 目前使用较多的是查表算法和 Cordic 算法. 查表算法具有精度高, 占用资源少的优点; Cordic 算法具有运算速度快的优势. 在具体应用过程中, 可以根据实际对时间和空间的要求灵活选择.

2 查表算法原理

查表法是一种快速实现非线性运算的最常用方法. 采用这种方法必须根据自变量的范围和精度要

求制作一张表格. 输入的范围越大, 精度要求越高, 占用存储量也越大. 把所选取的在自变量定义域内的区间称作制表区间, 用关键点将制表区间划分成若干份, 查找表中记录的只是制表区间内有限点的函数值. 通常情况下, 由于关键点的数量相对较多, 可以把两个关键点之间的曲线看作是线性的. 当自变量取到某两个关键点之间某点时, 通过两个关键点的函数值, 可以近似求出此点处的函数值. 通过这种方法, 可以计算出整个定义域内的函数值.

3 Cordic 算法原理^[1,2,4,5]

本文以圆周系统为对象, 通过旋转模式, 介绍 Cordic 算法的基本原理.

设一向量 (x'_1, y'_1) 经过旋转 θ 角得到另一个向

收稿日期: 2011-05-10; 修回日期: 2011-06-29

基金项目: 国家自然科学基金项目(60876017, 61006018)

量 (x'_2, y'_2) , 根据坐标变换规则, 有如下关系:

$$\begin{aligned} \begin{bmatrix} x'_2 \\ y'_2 \end{bmatrix} &= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{bmatrix} x'_1 \\ y'_1 \end{bmatrix} \\ &= \cos\theta \begin{pmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{pmatrix} \begin{bmatrix} x'_1 \\ y'_1 \end{bmatrix} \end{aligned} \quad (1)$$

将旋转角 θ 分解为 N 个递减的小旋转角 θ_i 之和, 即 $\theta = \sum_{i=0}^{N-1} \delta_i \theta_i$, 其中 $\theta_i \geq 0$, θ_i 顺时针旋转时 $\delta_i = -1$, 逆时针旋转时 $\delta_i = 1$. 对每次小的旋转

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \cos\theta_i \begin{pmatrix} 1 & -\delta_i \tan\theta_i \\ \delta_i \tan\theta_i & 1 \end{pmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (2)$$

$i=0, 1, 2, \dots, N-1$

令 $\theta_i = \arctan(2^{-i})$, 则 $\tan\theta_i = 2^{-i}$. 经过 N 次迭代可以得到:

$$\begin{bmatrix} x_N \\ y_N \end{bmatrix} = \prod_{i=0}^N \cos\theta_i \begin{pmatrix} 1 & -\delta_i 2^{-i} \\ \delta_i 2^{-i} & 1 \end{pmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (3)$$

由于 $N = +\infty$ 时, $\prod_{i=0}^N \cos\theta = 0.607253$, 为了跟踪已旋转的角度并判断 δ_i 的符号, 引入新变量 $Z_{i+1} = Z_i - \delta_i \arctan(2^{-i})$, 表示第 i 次旋转后剩余未旋转的角度. 假设初始旋转的角度为 $Z_0 = \theta$, 经过 N 次旋转后, 使得 $Z_N = 0$, 这时:

$$\delta_i = \begin{cases} -1 & \text{if } Z_i < 0 \\ 1 & \text{if } Z_i > 0 \end{cases}, i = 0, 1, 2, \dots, N-1$$

$$\begin{aligned} x_{i+1} &= x_i - \delta_i y_i 2^{-i} \\ y_{i+1} &= y_i + \delta_i x_i 2^{-i} \end{aligned} \quad (4)$$

经过 N 次迭代后,

$$\begin{aligned} x_N &= P[x_0 \cos Z_0 - y_0 \sin Z_0] \\ y_N &= P[y_0 \cos Z_0 + x_0 \sin Z_0] \\ Z_N &= 0 \\ P &= \frac{1}{0.607253} \end{aligned} \quad (5)$$

取 $x_0 = \frac{1}{P}, y_0 = 0, Z_0 = \theta$. 则有:

$$\begin{aligned} x_N &\rightarrow P[x_0 \cos Z_0 - y_0 \sin Z_0] = \cos\theta \\ y_N &\rightarrow P[y_0 \cos Z_0 + x_0 \sin Z_0] = \sin\theta \end{aligned} \quad (6)$$

由此可以得到任意输入角的正弦和余弦取值.

4 采用查表算法实现三角函数运算

4.1 算法实现

本文取 32 位定点小数, 小数点固定在最高位之后. 制表区间取为弧度值 $(0, 1.57)$ 内, 分成 201 组小区间, 每组长度为 0.0078125, 也就是 2^{-7} , 对应于二进制的 0.0000001. 每个小区间内部用 matlab 进

行了正弦函数在第一象限 $(0, 1.57)$ 内的线性拟合, 得到每个小区间内线性线段的斜率和截距值. 将每组斜率和截距值的 32 位定点小数表示制成 ROM 表. 对于每一个输入数值, 其前八位作为小区间的判定, 然后提取 ROM 中此小区间对应的斜率和截距值, 要得到的正弦函数就是此输入数值乘以斜率后与截距值相加的结果. 实现该部分关键代码如下:

```
assign product = radian * slope;
assign sum = product[62:31] + intercede;
if (enable)
    begin
        address <= radian[31:24];
    end
```

4.2 实验结果及分析

采用 Modelsim 6.5 软件对本设计进行时序仿真和功能验证. 采用 ISE 在 Virtex5 器件上进行综合, 该设计消耗 471 个 LUTs, 471 个寄存器, 最大路径延迟为 17.994ns, 最小频率为 55.574MHz. 在精度方面, 由于采用正弦函数拟合, 在零点附近斜率最大, 通过计算 $\cos 2^{-7}$ 的值, 可以得到最大误差为 0.00003052, 即可以至少精确到小数点后四位.

由此可以看出, 查表算法具有占用资源少, 精度高的优点, 但在实际应用过程中由于需要频繁从存储器中读取数据, 运算速度较慢. 所以在需要严格控制运算资源, 而对存储空间的要求不苛刻时, 查表算法是较好的选择.

5 采用 Cordic 定点算法实现三角函数运算

5.1 算法实现

对 32 位定点小数, 采用 Cordic 十级流水线迭代算法, 首先算出 $\arctan(2^{-i}), i=0, 2, \dots, 9$ 的值, 将其二进制定点小数表示作为每一级迭代时 Z 的修正参量. 然后将 x_0 赋初值 0.6037, y_0 赋初值 0, 之后每一级迭代皆为上一级的和差与移位操作, Z 的正负决定操作为相加或者是相减. 该部分关键代码如下(以第六级迭代为例):

```
if(z5[31])
    begin
        x6 <= x5 + {{5{y5[31]}}}, y5[31:5]};
        y6 <= y5 - {{5{x5[31]}}}, x5[31:5]};
        z6 <= z5 + 32';
        b0000001111111111010101010111000;
    end
else
```

```
begin
x6<=x5-{{5{y5[31]}},y5[31:5]};
y6<=y5+{{5{x5[31]}},x5[31:5]};
z6<=z5-32'
b0000001111111111010101010111000;
end
```

5.2 实验结果及分析

采用 Modelsim 6.5 软件对本设计进行时序仿真和功能验证,对于每一个输入的数值,输出在 10 个时钟周期之后产生,输出值与理论估算值一致.用 ISE 在 Virtex5 器件上进行综合,消耗 975 个 LUTs,995 个寄存器.最大频率为 359.195MHz.

以正弦值为例,比较其精度(精确到小数点后八位)如表 1 所示.

表 1 Cordic 算法运算结果与标准值的比较

角度	Cordic 模块结果 a	标准值 b	b-a
8	0.13840762	0.13917310	0.00076548
16	0.27487075	0.27563735	0.00076660
24	0.40358567	0.40673664	0.00315097
32	0.52680779	0.52991926	0.00311147
40	0.63881254	0.64278760	0.00397506
48	0.73881930	0.74314482	0.00432552
56	0.82466912	0.82903757	0.00436845
64	0.89385688	0.89879404	0.00493716
72	0.94574356	0.95105651	0.00531295
80	0.97997093	0.98480775	0.00483682

由此可以看出,采用 Cordic 十级流水线迭代法,可精确到小数点后两位,而要得到更高的精度,则需增加迭代次数,相应的计算资源消耗也会增加.两种算法的比较如表 2 所示.

表 2 两种算法的比较

算法选择	时钟频率 (MHz)	占用资源 (LUT)	精度(十进制 小数点后位数)
查表法	最小 55.574	471	4
Cordic 算法	最大 359.195	975	2

由此可见,定点 Cordic 设计具有速度快,精度易操作的特点.但是,相对于查表法设计,Cordic 设计的精度要低.

6 采用 Cordic 浮点算法实现三角函数运算

6.1 IEEE-754 标准浮点结构

IEEE — 754 标准有单精度、双精度和扩展精度三种浮点数据表示格式.本文使用 32 位单精度浮点小数进行运算,包括阶码 8 位,尾数 24 位(内含 1 位符号位).其数值范围 $3.4E-38\sim 3.4E+38^{[3-4]}$. 格式如表 3 所示(单精度中的指数位与尾数位之间隐含一个整数 1).

表 3 32 位浮点数据格式

符号位	指数位	尾数位
31	30-23	22-0

在某些数字芯片设计的实际应用中,仍然需要对这些浮点数进行三角函数运算.由于浮点定点的转换运算需要占用大量系统资源,直接的浮点数三角函数运算成为一种很好的选择.本文通过对代码的简单修改,利用 Cordic 算法的基本原理,实现了 32 位单精度浮点小数的三角函数运算.

6.2 算法实现

同样采用 Cordic 十级流水线迭代算法,首先算出 $\arctan(2^{-i}), i=0,2,\dots,9$ 的值,将其单精度浮点表示作为每一级迭代时 Z 的修正参量.然后将 x_0 赋初值 0.6037^[6], y_0 赋初值 0,之后每一级迭代皆为上一级的和差与指数位的减 1 操作,Z 的正负决定操作为相加或相减.该部分关键代码如下(以第六级迭代为例,fadd 为调用 Xilinx 库元件生成的 32 位浮点加运算单元):

```
wx5[31]<=z5[31];wx5[30:23]<=x5[30:23]-5;wx5[22:0]<=x5[22:0];
wy5[31]<=z5[31];wy5[30:23]<=y5[30:23]-5;wy5[22:0]<=y5[22:0];
wz5[31]<=~z5[31];// atan=0.895174',
tan=1/64,32'h3C7FFAAB
fadd X5(clk,x5,wx5,x6),Y5(clk,y5,wy5,y6),Z5(clk,z5,wz5,z6).
```

6.3 仿真与综合分析

采用 Modelsim 6.5 软件对本设计进行时序仿真和功能验证,对于每一个输入的数值,输出在 10 个时钟周期之后产生,输出的浮点值与理论估算值一致.采用 ISE 在 Virtex5 器件上进行综合,消耗 8428 个 LUT,11168 个寄存器,64 个 DSP48Es.最大频率为 342.349MHz.由于用到了浮点加运算,系

统资源消耗比传统定点查表或者定点 Cordic 算法大。

7 结束语

本文介绍了两种定点数三角函数在 FPGA 上的实现方法,并且比较了这两种方法在消耗系统资源,运算速度和精度方面的优劣。查表算法计算精度高,占用系统资源少;定点 Cordic 算法运算速度快,精度易操控。

采用 Cordic 算法直接设计 32 位浮点数三角函数是本文的新意所在。该设计在不经定点浮点转换的基础上,通过对浮点数指数位减 1 操作和浮点加运算实现了 Cordic 迭代算法,使 32 位浮点数的三角函数运算更加便捷。

参考文献:

- [1] Volder. The cordic trigonometric computing technique. IRETrans [J]. Electronic Computers, 1959, EC-8(3): 334-334.
- [2] 李全,陈石平,付佃华. 正交三角函数的 Cordic 实现 [J]. 微计算机信息, 2008, 24(3).
- [3] 马士超,王贞松. 基于 DSP 的三角函数快速计算 [J]. 2005, 31(22).
- [4] 张天瑜. 基于旋转模式的改进型 Cordic 算法研究 微电子学与计算机 2010(3): 93-97.
- [5] 王亚春,蔡德林,张梦龙,等. 流水线 Cordic 算法的 FPGA 实现 [J]. 通信技术, 2010(11): 169-171.
- [6] 张建斌,梁芳,刘乃安. 一种改进型算法的 Cordic 实现 [J]. 微电子学与计算机, 2010(11): 181-184.

作者简介:



吴庆达 男,(1984—),硕士研究生. 研究方向为数字集成电路设计。

何书专 男,工程师. 研究方向为数字集成电路设计。
潘红兵 男,副教授. 研究方向为数字集成电路设计。
沙金 男,副教授. 研究方向为数字集成电路设计。
李伟 男,工程师. 研究方向为数字集成电路设计。

(上接第 112 页)

(1) 粗糙集的不分辨关系是以 $\{A_n\}$ 行属性值对 D 分辨关系为基础,而算法是以 $\{A_n\}$ 中属性 A_i 对 D 分辨关系为基础。

(2) 算法用属性局部参数同值量和全局参数类别值共同确定属性分辨率,只涉及单一属性与决策属性的关系。由 S 直接得决策树,不需求等价类和属性约简等全局性的复杂运算。

(3) 从比较分析可知,一般在 $\{MIA_j\}$ 集中就只有一个必选属性;而当 $\{MIA_j\}$ 集中有多个必选属性时,采用降层的方法继续分辨,这就极大地提高了算法的精确性。

(4) 算法简洁易理解;时间和空间复杂度极小。

参考文献:

- [1] 丁春荣. 基于粗糙集的决策树构造算法 [J]. 计算机工程, 2010, 36(11): 75-77.
- [2] 李永华. 一种基于 rough 集的属性约简的改进算法 [J]. 计算机应用, 2008, 28(8): 200-202.
- [3] 王杨. 基于小生境遗传算法的粗糙集属性约简方法 [J]. 计算机工程, 2008, 34(5): 66-70.
- [4] 高静. 一种改进的基于正区域的决策树算法 [J]. 计算机科学, 2008, 35(5): 138-142.
- [5] 陈婷. 基于粒度的粗集-决策树雷达信号识别模型 [J]. 微电子学与计算机, 2008, 25(12): 13-16.

作者简介:

刘军 女,(1977—),讲师. 研究方向为数据挖掘、粗糙集、粒计算。