

## 基于嵌入式的 CORDIC 算法的改进及实现

李美俊, 李光明

(陕西科技大学 电气与信息工程学院, 陕西 西安 710021)

**摘 要:** 介绍了 CORDIC 算法的基本原理, 分析了 CORDIC 算法的具体计算方法. 针对利用 CORDIC 流水线实现 FFT 蝶形运算耗费资源多的问题, 依据 CORDIC 计算迭代系数的方法改进了 CORDIC 流水线的结构形式, 使其适应 FFT 算法. 整个 FFT 处理器的实现主要利用了 CycloneII 系列的 EP2C35F672C6. 并通过时序仿真和硬件仿真来进行比较. 它们的计算结果基本一致.

**关键词:** CORDIC; FFT; Cyclone II; 流水线

**中图分类号:** TP274

**文献标识码:** A

**文章编号:** 1000-7180(2012)02-0142-04

## The Improvement and Implementation of CORDIC Algorithm Based on Embedded System

LI Mei-jun, LI Guang-ming

(School of Electric and Communication Engineering, Shaanxi University of Science and Technology;  
Xi'an 710021, China)

**Abstract:** Introduces the basic principle of CORDIC algorithm, and analyses the calculation method of CORDIC algorithm. For use of CORDIC streamline realize FFT wing operation cost more resources problems, according to CORDIC iterative method of calculating coefficient to improve CORDIC pipeline structure form, in order to accommodate FFT algorithm. Choose ALTERA company CycloneII series of EP2C35F672C6 to realize the whole FFT processor. And timing simulation and hardware emulation are designed. Through comparing the both, the calculation results are basically the same.

**Key words:** CORDIC; FFT; Cyclone II; Streamline

### 1 引言

CORDIC 算法(The Coordinate Rotational Digital Computer, 坐标旋转计算机)是一种循环的迭代算法, 它的主要思想是用一些基本运算函数和算术操作来运算基数相关的角度, 从而使得逐渐逼近所需旋转的角度, 从根本上来说它是利用数值性计算逼近的方法来实现的, 在传统上来说应该用乘除法来实现, 但是这样需要消耗的硬件资源大, 并且实现起来也有很大的困难, 为了解决这个问题, 所以 CORDIC 算法便随之产生了, 它仅仅与固定的角度与计算基数有关, 运算只有移位和加/减. 这样就可以利用简单的硬件电路来实现它的移位操作了, 从

而大大节约了硬件的资源, 使得 CORDIC 算法在硬件上可以得到较好的实现. 根据它的迭代原理, CORDIC 单元可以用流水线结构进行表示, 可以使向量旋转并行处理, 这样在很大程度上加快了蝶形运算的速度<sup>[1]</sup>. 但是这样的迭代如果太多也会占用大量的芯片资源, 特别是在使用多个蝶形进行 FFT 处理时, 这里通过对 CORDIC 流水线结构的改进, 来降低芯片资源.

### 2 CORDIC 算法原理

CORDIC 算法在起初并没有发挥它应有功能, 而是被用作实现二进制和十进制的转换工具. 这样持续了整个六十年代, 直到 1971 年随着 VLSI 技术

的不断发展,人们才开始进一步的对他进行了研究,并此时 Walther 提出了统一的 CORDIC 算法,CORDIC 算法才展示出广泛的应用前景<sup>[2]</sup>. 在如今 CORDIC 算法已被广泛应用,诸如离散傅里叶变换、矩阵的分解、矩阵特征值的求解、场分解、线性预测参数的求解等,在现代信号处理各种算法实现中起到了重要的作用.

如图 1 所示,点  $M$  的坐标从  $(x_0, y_0)$  通过直角坐标轴顺时针旋转角度  $A$ ,变为  $(x, y)$ .

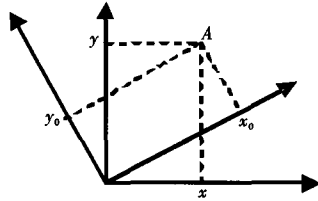


图 1 坐标旋转

旋转方程可表示为

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos A & -\sin A \\ \sin A & \cos A \end{bmatrix} \begin{bmatrix} x(0) \\ y(0) \end{bmatrix} \quad (1)$$

旋转角度  $A$  的实现可通过把这个角度分解成若干个角度来实现,每一个角度可以作为一步,这样就可以每一步旋转一个较小的角,从而完成旋转的一部分.

单独的一步的旋转方程可表示为

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos A_n \begin{bmatrix} 1 & -\tan A_n \\ \tan A_n & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \quad (2)$$

若将旋转角度分为  $n$  分,则  $A_n$  之和为  $A$ ,并且可定义顺时针为正角度,这里以  $\theta_n$  以表示角度的大小,  $S_n$  表示符号的正负,那么  $A_n$  则就可表示为  $S_n A_n$ . 所以每一步要旋转的角度可以表示为

$$\theta_n = \arctan\left(\frac{1}{2^n}\right) \quad (3)$$

需旋转的角度  $A$  则可以表示为每一步小角度之和:

$$\sum_{n=0}^{\infty} A_n = \sum_{n=0}^{\infty} S_n \theta_n = A \quad (4)$$

式中,  $S_n$  表示的是正角度或负角度.  $S_n = -1$  表示负角度及第  $n$  步坐标轴要逆时针旋转,  $S_n = +1$  则表示正角度及顺时针旋转. 当  $\tan A_n$  的值等于  $S_n 2^{-n}$ . 则每一步的旋转方程表示为

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos A_n \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \quad (5)$$

上式除  $\cos A_n$  外,整个算法就只是移位和加法运算了. 由上式知

$$\cos A_n = \frac{1}{1 + S_n^2 2^{-2n}} \quad (6)$$

当  $S_n \neq 0$ , 方程可表示为

$$\cos A_n = \frac{1}{1 + 2^{-2n}} \quad (7)$$

从上式可以看出,  $\cos A_n$  是与角度无关的一个值,多个  $\cos A_n$  的乘积可作为一个常数  $P(N)$ ,如果设这个常数的倒数为  $K(N)$ ,则有如下表达式:

$$K(N) = \frac{1}{P(N)} = \prod_{n=0}^{N-1} \frac{1}{\cos A_n} = \prod_{n=0}^{N-1} (1 + 2^{-2n}) \quad (8)$$

当  $N=13$  时  $P(N) \approx 0.607\ 253$ . 这时  $\cos A_n$  可以从方程中提出,既表达式(5)可以简化为如下:

$$\begin{aligned} X_{n+1} &= X_n - S_n Y_n * 2^{-n} \\ Y_{n+1} &= X_n + S_n X_n * 2^{-n} \end{aligned} \quad (9)$$

在通过有限次迭代,得到  $X_N$  和  $Y_N$  的表达式如下:

$$\begin{aligned} X_N &= K(N) (X_0 \cos(Z_0) - Y_0 \sin(Z_0)) \\ Y_N &= K(N) (Y_0 \cos(Z_0) + X_0 \sin(Z_0)) \end{aligned} \quad (10)$$

此时旋转的坐标值可以表示为

$$\begin{aligned} X &= X_N * P(N) \\ Y &= Y_N * P(N) \end{aligned} \quad (11)$$

令  $Z_0 = \theta$ ,  $Y_0 = (P_{re} - Q_{re})$ ,  $X_0 = (P_{im} - Q_{im})$  得

$$\begin{aligned} X_k &= k(N) ((P_{im} - Q_{im}) \cos(\theta) - (P_{re} - Q_{re}) \sin(\theta)) \\ Y_k &= K(N) ((P_{re} - Q_{re}) \cos(\theta) - (P_{im} - Q_{im}) \sin(\theta)) \end{aligned} \quad (12)$$

上式中  $K_1$  为旋转增益的极限值,因为

$$K(N) = \prod_{n=0}^{N-1} \frac{1}{\cos A_n} = \prod_{n=0}^{N-1} (1 + 2^{-2n}) \quad (13)$$

通过求极限可以求出  $K_1 \approx 1.646\ 759 > 1$ , 为了防止数据溢出,利用 CORDIC 算法的计算结果右移一位(即乘以 0.5).

$$\begin{aligned} R_{re} &= K(N)' (P_{re} + Q_{re}) \\ R_{im} &= K(N)' (P_{im} + Q_{im}) \\ S_{re} &= K(N)' ((P_{re} - Q_{re}) \cos(\theta) + (P_{im} - Q_{im}) \sin(\theta)) \\ S_{im} &= -K(N)' ((P_{re} - Q_{re}) \sin(\theta) + (P_{im} - Q_{im}) \cos(\theta)) \end{aligned} \quad (14)$$

其中  $K(N)' = K(N)/2$ .

### 3 CORDIC 的流水线结构及其改进

在这里的流水线结构中,由于旋转角度集的值连接到了累加器上,并且每次的移动都是固定的深

度,不需要太多的存储空间,当执行其他进程时,它还可以输入数据,从而使得整个系统的运行效率提高,具体每个单元的连接,如图 2 所示。

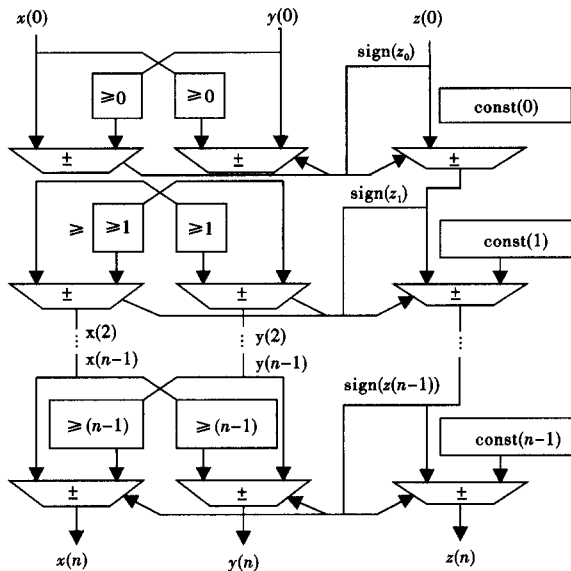


图 2 旋转模式下水流线的结构

为了提高速度来满足 FFT 的要求,CORDIC 必须设计成流水线的形式来提高速度。为了能够获取所需要的值,需要在  $Z_0$  数据通道里加入需要旋转的角度,这时便可以通过  $Z_i$  与固定角度的运算来取得需要的值。此时还需要旋转向量在各级迭代中的旋转方向来处理角度的正负效果。通过多次迭代,  $Z_n$  趋近于零,向量旋转到相应角度。为了实现 FFT 的蝶形单元中用 CORDIC 代替复乘单元,可以将数据的实部和虚部添加到  $x_0$  和  $y_0$  通道,并且为了实现乘以的目的需要将系数作为角度从  $Z_0$  处输入。

为了更好地实现 CORDIC 算法在 FFT 上的应用,这里对流水线形式进行改进,如图 3 所示,这里可以在  $X_0$  和  $Y_0$  数据通道里分别添加向量的实部

和虚部,为了控制向量在各级迭代中旋转的方向,他们的系数需要输入到 D 触发器中和向量保持同步。向量经多次迭代旋转到相应角度<sup>[3]</sup>。

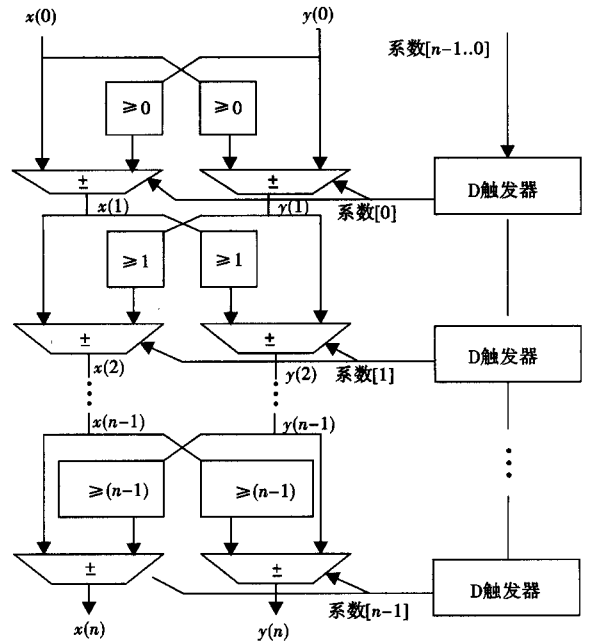


图 3 改进的流水线结构

#### 4 CORDIC 的旋转系数

为了能够求出 CORDIC 的旋转系数,需要利用 MATLAB 语言编写程序求出各级旋转系数,并将系数存在 ROM 里,它主要是通过迭代原理来计算的,当计算出旋转系数时,就可以按照改进后 CORDIC 的结构来进行了,时序仿真结果如图 4 所示。

图中计算的是利用改进的 CORDIC 算法计算的结果。从仿真图中可以看到:为了计算出相应的结果,必须要一个合适的的迭代系数  $C$ 。

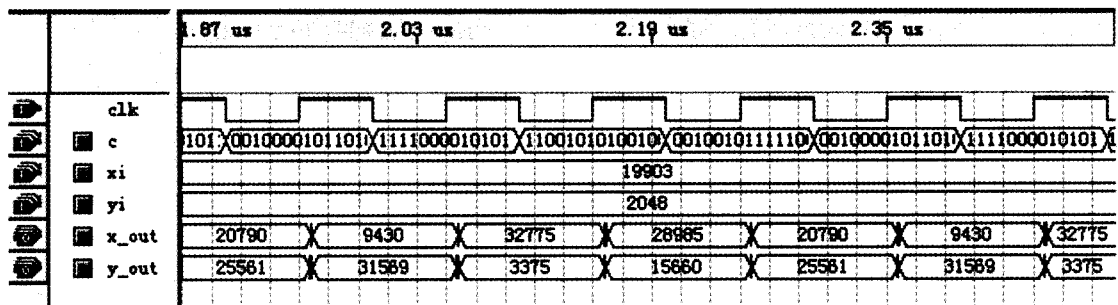


图 4 CORDIC 算法仿真图

## 5 FFT 处理器的仿真和测试

这里主要通过 ALTERA 公司 CycloneII 系列的 EP2C35F672C6 来实现整个 FFT 处理器,并且利用它进行了仿真和测试.

### (1) 直流信号的测试

首先利用 MATLAB 来产生一个直流信号: $i=1:256, x(i)=50$ ; MATLAB 计算的直流信号 FFT 结果  $X(0)=14\ 000$ , FFT 处理器输出的结果:  $X(0)=6\ 000$ . 并且它们的虚部都为 0, 它们的结果是一样的, 虽然 CORDIC 算法使运算使每一级的结果扩大了 1.65 倍, 但是为了减少误差和防止溢出在 FFT 处理器的各级都进行了  $1/2$  的截尾处理. 因此这样的处理不会影响频谱的显示.

### (2) 三角波信号的测试

首先利用 MATLAB 产生一个三角波信号: $i=1:128, x(i)=i; (28+i)=128$ . 图 5 和图 6 所示为 MATLAB 的计算结果, 图 7 和图 8 所示为 FFT 处理器计算结果. 为了更好地说明计算结果的一致性, 这里将 FFT 处理的结果进行了还原 (FFT 处理器对各级进行了放大和截尾), 这里通过图对实部和虚部进行了详细的比较, 可以证明计算结果基本一致.

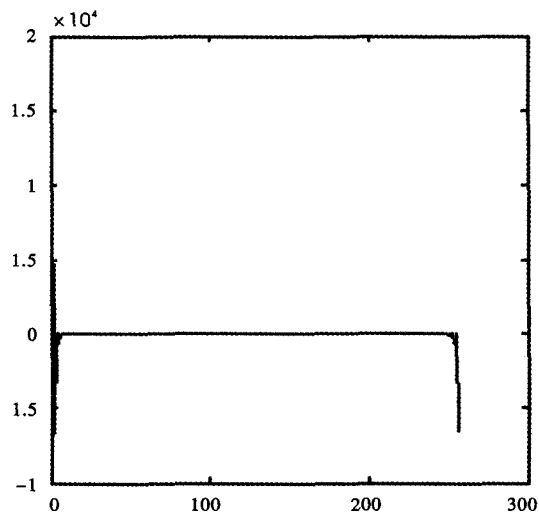


图 5 MATLAB 实部计算结果

## 6 结束语

依据 CORDIC 计算迭代系数的方法改进了 CORDIC 流水线的结构形式, 采用 CORDIC 算法以较少的资源实现了快速乘法器, 使其适应 FFT 算法. 整个 FFT 处理器的实现主要利用了 CycloneII 系列的 EP2C35F672C6, 并通过时序仿真和硬件仿真

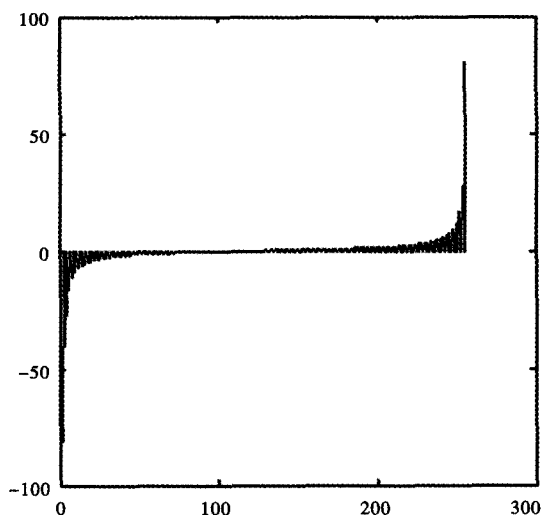


图 6 MATLAB 虚部计算结果

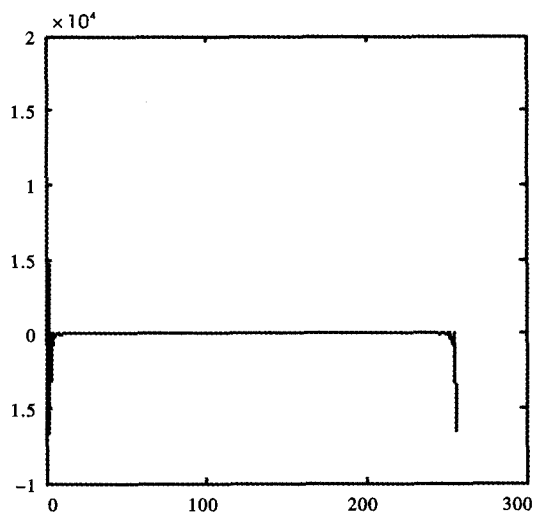


图 7 FFT 实部计算结果

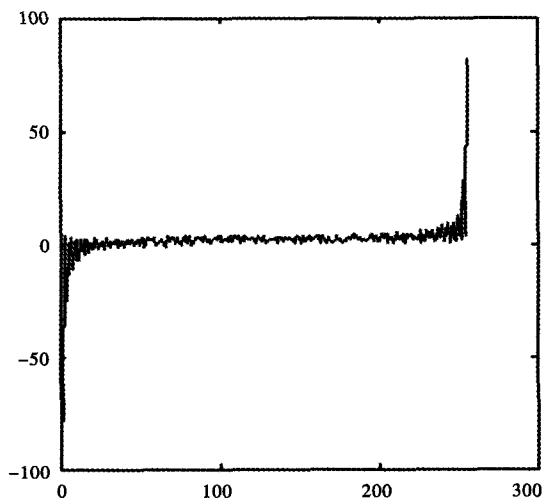


图 8 FFT 虚部计算结果

(下转第 149 页)

通过实验对比分析,改进后的 Canny 边缘检测算法,在噪声抑制、边缘细节保持方面优于传统的 Canny 算子,而且还能够有效减弱伪边缘现象。

## 6 结束语

本文对传统 Canny 边缘检测算子存在的缺陷,通过分析,用形态学平滑替代经典 Canny 算子中的高斯滤波,使得图像更加平滑,同时保留更多的细节信息;利用 Otsu 方法对阈值进行自适应设定,理论和实验证明,该方法确定的双阈值可以边面人为干扰。检测出的边缘更加连续并减少虚假边缘的存在。提高了传统的 Canny 算子的应用范围和自动性能。

### 参考文献:

- [1] 杜文亮,刘晓东,吕园园. 基于 C-Canny 算子与灰度空间的彩色图像边缘检测 [J]. 微电子学与计算机, 2010 (4): 17-20.
- [2] 章毓晋. 图像分割 [M]. 北京: 科学出版社, 2001: 135-156.

- [3] 姜兴乾,李星野,贾淑华. 基于最大熵与小波变换的图像边缘检测算法 [J]. 微电子学与计算机, 2009 (3): 189-191.
- [4] Rafael C. Gonzalez, Richard E. Woods. 数字图像处理 [M]. 北京: 电子工业出版社, 2005.
- [5] 彭辉,文友先,吴兰兰,等. 采用自适应 Canny 算子的树上柑橘图像边缘检测 [J]. 计算机工程与应用, 2011 (47): 163-166.
- [6] 王佐成,刘晓冬,薛丽霞. Canny 算子边缘检测的一种改进方法 [J]. 计算机工程与应用, 2010 (46): 202-205.
- [7] Zhang Jun, Hu Junlu. Image segmentation based on 2D Otsu method with histogram analysis [C]// International Conference on Computer Science and Software Engineering. [s. l]: IEEE, 2008: 105-108.

### 作者简介

韩慧妍 女, (1980—), 博士研究生. 研究方向为信号与信息处理。

(上接第 145 页)

来进行比,可以看出,解决了 FFT 蝶形运算耗费资源多的问题。

### 参考文献:

- [1] 胡国荣,孙允恭. CORDIC 算法及其应用 [J]. 信号处理, 1991, 12: 229-242.
- [2] Volder J E. The CORDIC trigonometric computing technique [J]. IRE T Electron Computers, 1959 (9): 330-334.

- [3] 于效宇,宋立新,刘艳. CORDIC 流水线结构在 FFT 设计中的改进 [J]. 哈尔滨理工大学学报, 2005, 10 (1): 55-57.

### 作者简介:

李美俊 男, (1983—), 硕士研究生. 研究方向为 ARM 技术的研究。

李光明 男, (1963—), 教授, 硕士生导师. 研究方向为计算机控制与计算机网络。