基于 FPGA 和改进 CORDIC 算法的 NCO 设计与实现

彭 奇，习友宝

（电子科技大学，四川 成都 611731）

摘 要：将 CORDIC 算法传统实现中的象限转换从输出转移到输入进行处理，简化了电路逻辑。针对 CORDIC 算法的流水线结构做出增大映射分区的改进，省去了流水线的第一级，减少了流水线结构内旋转角的一位数据宽度。在 FPGA 中仿真并实现了基于该改进 CORDIC 算法的 NCO。仿真结果有良好的精度，证明了该方案的可行性。

关键词：FPGA；改进 CORDIC；NCO

中图分类号：TN75 **文献标识码**：A **文章编号**：1674-7720(2013)05-0060-03

Design and implementation of NCO based on FPGA and improved CORDIC algorithm

Peng Qi, Xi Youbao

(University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: Do the quadrant transformations of CORDIC algorithm with input data instead of output data used in the classical method and simplify the circuit. Meanwhile, improve the pipelined CORDIC algorithm by increasing it's mapping partitions, so it can discard the first stair of the pipeline series and a bit of the data about rotation angle in the pipeline structure. Simulate and implement the NCO based on FPGA and the improved CORDIC algorithm. It proves the feasibility of the scheme with the precision simulation results.

Key words: FPGA; improved CORDIC; NCO

数控振荡器 NCO（Numerically Controlled Oscillator）是基于数值控制输出特定波形的装置或电路内核，是直接数字频率合成（DDS）和各种数字信号发生器的核心部分。NCO 广泛应用于数字通信中的调制解调单元，其产生的正弦信号可用作 FSK、BPSK 等多种调制方式的载波。NCO 通常可用计算法和查表法来实现。两种方法相比较，查表法的逻辑控制单元比计算法简单，工作频率更高，但其所占用内部存储单元比计算法更多，造成在存储资源紧张的系统中难以实现。同时，由于查表法耗费大量的 ROM 资源，因而会增大器件能耗。而随着可编程逻辑器件的发展，高精度、高速度的实时计算是可以实现的。本文介绍在 FPGA 中用改进的坐标旋转数字式计算机 CORDIC(Coordinate Rotation Digital Computer)算法来实现 NCO 的方法。

1 CORDIC 算法

CORDIC 算法的基本思想是用一系列固定的、与运算基数相关的角度不断偏摆从而逼近所需旋转的角度。从广义上讲它是提供一个数值性计算的逼近方法，由于这些固定的角度与计算基数有关，运算只有移位和加减[1]。

如图 1 所示，直角坐标系中的单位向量 $A(x_1,y_1)$ 旋转角度 θ 到单位向量 $B(x_2,y_2)$ 可以由式(1)表示。

$$\begin{cases} x_2=x_1\cos\theta-y_1\sin\theta=\cos\theta(x_1-y_1\tan\theta) \\ y_2=x_1\sin\theta+y_1\cos\theta=\cos\theta(y_1+x_1\tan\theta) \end{cases} \quad (1)$$

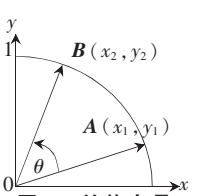

图 1 单位向量的平面旋转

CORDIC 算法核心是旋转角度 θ。设第 n 次旋转角度为 $\theta_n(n=0,1,2,\cdots)$，且有 $\tan\theta_n=2^{-n}$，则总的迭代旋转可表示为：

$$\begin{cases} x_{n+1}=K_n(x_n-d_n2^{-n}y_n) \\ y_{n+1}=K_n(y_n+d_n2^{-n}x_n) \\ z_{n+1}=z_n-d_n\theta_n \end{cases} \quad (2)$$

其中，$K_n=\cos\theta_n=1/\sqrt{1+2^{-2n}}$；第三个方程为引入的角度累加器，用于跟踪累加的旋转角度，z_0 表示初始角度；$n=$

0,1,2…为旋转次数；$d_n=\text{sign}(z_n)$为判决算子，用于确定旋转方向（当 $d_n=1$ 时，逆时针旋转；当 $d_n=-1$ 时，顺时针旋转）。

由 $\theta_n=\arctan 2^{-n}(n=0,1,2\cdots)$ 可知，每次旋转的角度都是确定的，且呈逐渐收敛的趋势，因而 K_n 也是确定的。又由于有式(3)：

$$\sum_{i=0}^{\infty}\arctan 2^{-i}\approx 99.883^\circ \tag{3}$$

所以位于$[-\pi/2,\pi/2]$的任意角度，都可以从 X 轴出发，经过一系列角度 $\theta_n=\arctan 2^{-n}(n=0,1,2\cdots)$ 的旋转到达。

经过 $n\to\infty$ 次迭代后，式(3)变为：

$$\begin{cases}x_n=A_n(x_0\cos z_0-y_0\sin z_0)\\ y_n=A_n(y_0\cos z_0+x_0\sin z_0)\\ z_n=0\end{cases} \tag{4}$$

其中，A_n 为伸缩因子，表示旋转后模值变为原来的 A_n 倍。A_n 如式(5)：

$$A_n=\prod_{i=0}^{n-1}K_i=\prod_{i=0}^{n-1}\sqrt{1+2^{-2i}}\approx 1.646\ 760\ 24\ (n\to\infty) \tag{5}$$

由式(4)可知，若令输入 $x_0=1/A_n$，$y_0=0$，则可得 z_0 的正余弦值。

CORDIC 算法有两种实现方法，一种是迭代法，由于迭代算法存在精度与速度相制约的问题，且每次迭代需要查一次存放 θ_n 的 ROM 表，所以本文采用第二种方法，即流水线法。

流水线结构的 CORDIC 算法在经过初始延迟之后，每一个周期输出一组数据，增加流水线级数不会影响到运行速度，流水线级数与运算精度成正比。而且 θ_n 的值可做为常数直接放在每一级，省去查找 ROM 表。流水线结构如图 2 所示。

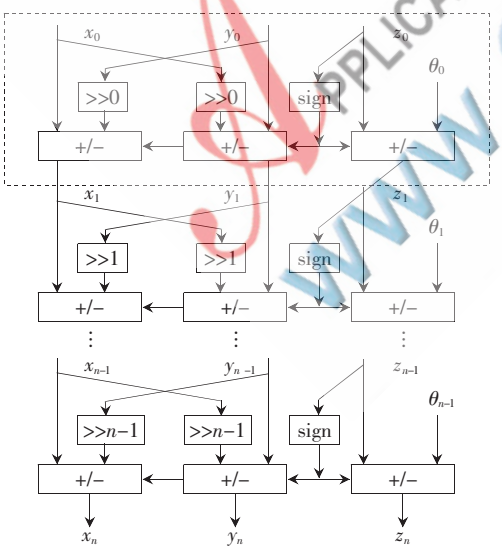

图 2 CORDIC 算法的流水线结构

由于 CORDIC 算法的计算范围为$[-\pi/2,\pi/2]$，所以需要利用正余弦的象限对称性将其映射到$[0,2\pi)$。常用的映射方法是将$[0,2\pi)$扩大到$[0,2^n)$。低 $n-2$ 位作为 CORDIC 算法的输入。高两位确定所在象限，用于对 CORDIC 算法的输出结果按照象限对称性转换得到最终结果。其结构框图如图 3 所示，其中 $x_0=1/A_n$，$y_0=0$。

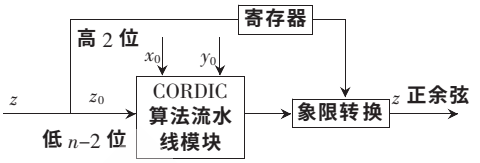

图 3 扩展后的正余弦计算框图

2 CORDIC 算法的改进

图 3 中的象限变换是加在流水线之后，这就相当于给整体通道多加了一级运算。如要计算第二象限角度 $z=\theta+\pi/2$（$\theta\in[0,2\pi)$）的余弦值，CORDIC 输入为 $z_0=\theta$，由 $\cos z=-\sin\theta=-\sin z_0$ 知，需选取 y_n 的输出并求补得到。这就给系统的实现增加了复杂度。

经过分析式(4)得到，可以将象限的转换通过控制 CORDIC 算法的不同 x_0 和 y_0 输入值实现。不同象限的输入见表 1。

表 1 四分圆映射表

输入 z 的范围	输入 x_0	输入 y_0
$[0,\pi/2)$	$1/A_n$	0
$[\pi/2,\pi)$	0	$1/A_n$
$[\pi,3\pi/2)$	$-1/A_n$	0
$[3\pi/2,2\pi)$	0	$-1/A_n$

输入 x_0 和 y_0 的选择通过角度 Z 的高两位控制两个四选一选择器就可简单地实现。

另外，由正余弦函数性质可知，正余弦函数不仅关于象限对称，还关于同一象限的上下两部分对称。因此，可以将旋转圆周再细分为八个区域，即四个象限再分为上部分和下部分。此时，用角度 z 的高三位进行区域转换，低 $n-3$ 位作为 CORDIC 的输入，则有 $z_0\in[0,\pi/4)$[2]。

又由：

$$\sum_{i=0}^{\infty}\arctan 2^{-i}\approx 54.883^\circ \tag{6}$$

知 CORDIC 流水线输入为 $z_0\in[0,\pi/4)$ 时，不用经过 45° 旋转，就可以使 $z_n=0$，从而完成算法。

所以八分圆映射可以省去流水线中的第一级，即图 1 中的虚线框部分，并且流水线中的 z_n 数据宽度减小了一位。此时，伸缩因子 A_n 应用式(7)所示 A'_n 替换。其输入映射如表 2 所示。

$$A'_n=\prod_{i=0}^{n-1}K_i=\prod_{i=0}^{n-1}\sqrt{1+2^{-2i}}\approx 1.644\ 435\ 33\ (n\to\infty) \tag{7}$$

改进的 CORDIC 算法框图如图 4 中虚线框内所示。

3 NCO 实现及仿真[4-5]

由以上的改进 CORDIC 算法加上相位累加器和初始相位控制单元就构成了基于 CORIC 算法的 NCO，如图 4 所示。其工作原理为：相位累加器的初始值为 0，频率控

表 2 八分圆映射表

输入 z 的范围	输入 x_0	输入 y_0
[0,π/4)	$1/A'_n$	0
[π/4,π/2)	$\sqrt{2}/(2A'_n)$	$\sqrt{2}/(2A'_n)$
[π/2,3π/4)	0	$1/A'_n$
[3π/4,π)	$-\sqrt{2}/(2A'_n)$	$\sqrt{2}/(2A'_n)$
[π,5π/4)	$-1/A'_n$	0
[5π/4,3π/2)	$-\sqrt{2}/(2A'_n)$	$-\sqrt{2}/(2A'_n)$
[3π/2,7π/4)	0	$-1/A'_n$
[7π/4,2π)	$\sqrt{2}/(2A'_n)$	$-\sqrt{2}/(2A'_n)$

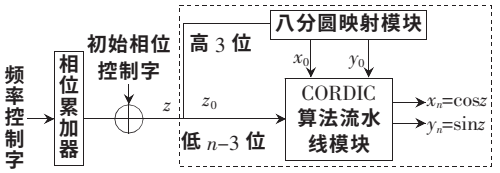

图 4 NCO 的实现框图

制字在每一个时钟周期与相位累加器累加一次，频率控制字的大小就表示相位的步长，得到的相位值再与相位控制字相加，得到当前相位值，送到改进 CORDIC 算法中进行计算，将相位值转换为幅度值。通过设置频率控制字改变输出正弦的频率，可用于调频。通过设置初始相位控制字改变输出正弦的初始相位，可用于调相。另外，若在 CORDIC 算法后再加一个乘法器，即可实现调幅，本文不考虑此项。

若设频率控制字为 K，则输出信号的频率为 $f=K\times f_c/2^N$，其中 f_c 为参考时钟，N 为相位累加器宽度。频率的最小分辨率 $f=f_c/2^N$，亦即最小输出频率。随着输出频率的增加，输出信号的采样点数会减少，从而会影响到波形的平滑度，所以在实际应用中，K 的值不宜过大。

本文给定相位宽度为 22 位，即 2^{22} 代表 360°，输出数据宽度为 18 位，其中最高位作为符号位，用于数值正负的判断。由于正余弦函数的输出总在[0,1]范围内，所以可以用 2^{16} 代表 1，则数据精度为 16 位。$(2^{16},2^{17})$ 和 $(-2^{16},-2^{17})$ 作为数据误差引起的溢出保护区[3]。设定 K 的 22 位数据宽度的低 14 位有效，高 8 位由常量 0 来代替。则该 NCO 能实现的最高输出频率约为 $f_{max}=f_c/2^8$。

本 NCO 基于 Altera 公司的 EP1C6Q240C8 FPGA 芯片和 Quartus II 平台进行设计和仿真。该设计采用 16 级流水线，共用到 50 个加法器、16 个非门、2 个八选一选择器和 6 个常量表达式。该 NCO 直接采用 Quartus II 中的 lpm_add_sub、lpm_mux 和 lpm_constant 逻辑单元模块进行原理图设计。CORDIC 流水线中的移位器可通过信号线的错位连接实现。$d_n=\text{sign}(z_n)$ 函数直接取 z_n 的符号位实现。

为对改进 CORDIC 算法的精度进行检验，不失一般性地任取一系列角度输入图 4 中的虚线框部分进行仿真。仿真结果数据分析如表 3。可见，设计的 NCO 是有较高精度的。

表 3 改进 CORDIC 算法的仿真结果误差分析

正余弦/(°)	精确值	仿真结果	误差
cos0	1	1.000 02	0.000 02
sin0	0	−0.000 05	−0.000 05
cos20	0.939 69	0.939 65	−0.000 04
sin20	0.342 02	0.342 07	0.000 05
cos56	0.559 19	0.559 17	−0.000 02
sin56	0.829 04	0.829 06	0.000 02
cos90	0	0.000 03	0.000 03
sin90	1	1.000 02	0.000 02
cos98	−0.139 17	−0.139 13	0.000 04
sin98	0.990 27	0.990 22	−0.000 05
cos180	−1	−1.000 03	−0.000 03
sin180	0	0.000 03	0.000 03
cos220	−0.766 04	−0.765 99	0.000 05
sin220	−0.642 79	−0.642 85	−0.000 06
cos300	0.5	0.499 98	−0.000 02
sin300	−0.866 03	−0.866 04	−0.000 01

由仿真结果可知，改进的 CORDIC 算法能够实现一定精度的正弦计算。另外，可通过拓宽数据宽度和增加流水线级数来增大算法精度。由改进 CORDIC 算法实现的 NCO 电路得到简化，性能稳定，能够满足 DDS、数字调制等相关应用。

参考文献

[1] 胡国荣，孙允恭.CORDIC 算法及其应用[J].信号处理，1991，7(4)：229-242.

[2] 张科峰，彭帅，蔡梦.基于 CORDIC 算法的 NCO[J].现代雷达，2008，30(6)：91-94.

[3] 王威. 高精度正余函数的 FPGA 实现 [J]. 电子科技，2011，24(1)：28-31.

[4] 王诚，蔡海宁，吴继华.ALtera FPGA/CPLD 设计(基础篇)(第 2 版)[M].北京：人民邮电出版社，2011.

[5] 杨小牛，楼才义，徐建良.软件无线电原理与应用[M].北京：电子工业出版社，2001.

(收稿日期：2013-01-07)

作者简介：

彭奇，男，1989 年生，在读研究生，主要研究方向：集成系统芯片(SoC)设计与工具开发。

习友宝，男，1964 年生，教授，硕士生导师，主要研究方向：集成系统芯片(SoC)设计与工具开发。