doi: 10.7690/bgzdh.2014.03.014

基于查找表和 CORDIC 算法的混合型 NCO 设计

李坤贺，岳曾敬
（中国兵器工业第五八研究所特种电子技术部，四川 绵阳 621000）

摘要：针对常用的在 FPGA 上实现 NCO 的查找表法和 CORDIC 算法存在的不足，提出一种基于查找表和 CORDIC 算法相结合的 NCO 实现方法。对查找表法、CORDIC 算法以及查找表和 CORDIC 算法相结合方法的实现原理进行设计，并分别对 3 种不同实现方式进行仿真和对比分析。结果表明：用查找法和 CORDIC 算法相结合的方法实现的 NCO 能很好地平衡 FPGA 上存储器资源和逻辑资源的消耗，并在产生信号的精度、速度、纯度方面也有很好的兼顾。

关键词：查找表；CORDIC 算法；混合型 NCO；FPGA
中图分类号：TP301.6　　**文献标志码：**A

Design of Hybrid NCO Based on Look-up Table and CORDIC Algorithm

Li Kunhe, Yue Zengjing
(*Department of Special Electronic Technology*, *Research Institute of China Ordnance Industries*, *Mianyang* 621000, *China*)

Abstract: Aiming at the shortcoming of NCO look-up table method and CORDIC method in FPGA. Put forwards a NCO realization method which combined look-up table method with CORDIC method. Design realization principle of look-up table method, CORDIC method and method which combined look-up table method with CORDIC method. Then carry out simulation and comparison analysis of three methods. The results show that the NCO that is achieved by the combination of the look-up table and the algorithm of CORDIC can well balance the consumption of the memory resources and the logical resources and it can give consideration to the accuracy, speed, and purity of the signals it produces.

Keywords: look-up table; CORDIC algorithm; hybrid NCO; FPGA

0　引言

数控振荡器（numerical controlled oscillator，NCO）由于产生的信号频率精度高、转换速度快、频谱纯度高、可控性好而被广泛应用于数字上、下变频和其他相位、频率调制解调系统中[1-2]。NCO 的目标就是产生一个频率可变的正弦或余弦信号：$S(n)=\sin(\omega_c n)=\sin\left(\frac{2\pi n f_c}{f_s}\right)$。其中，$f_c$ 为本地振荡频率即 NCO 的输出频率；f_s 为 NCO 的时钟频率即输入信号的采样频率，$n=0,1,2,3,\cdots$ [3-4]。

目前在 FPGA 上实现 NCO 的常用方法有查找表法和 CORDIC 算法。但是要在 FPGA 上实现高精度的 NCO，单纯用查找表法会占用较多的存储器资源，单纯用 CORDIC 算法又会消耗大量的逻辑资源。基于此，笔者提出一种基于查找法和 CORDIC 算法相结合的 NCO 实现方法，以减少资源消耗，兼顾产生信号的精度、速度、纯度。

1　基于查找表法的 NCO 实现原理

基于查找表法实现的 NCO 主要由相位累加器、相位加法器和查找表 3 部分组成。其结构如图 1。

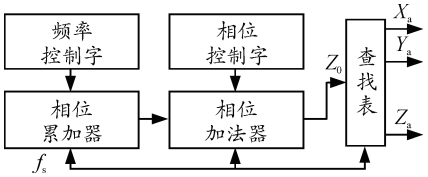


图 1　查找表法实现 NCO 的工作原理

其中，相位累加器在系统时钟 f_s 作用下以频率控制字为步进进行累加，将相位累加器的输出和相位控制字在相位加法器中相加得到的结果作为查找表的地址，通过在查找表中查找得到预存在其中的函数值产生期望的信号。该方法实现方便、简单，所需要的逻辑资源很少，信号输出速度快、纯度高，但精度主要依赖于查找表的深度和存储数据的宽度，假设希望输出信号的相位角度误差小于 $\arctan(2^{-15})$，那么查找表的深度就需要达到 256 K

收稿日期：2013-11-10；**修回日期：**2014-01-14
作者简介：李坤贺（1987—），男，河南人，硕士研究生，从事嵌入式计算机技术研究。

个样点，即使采用象限对称压缩技术也还需要 64 K 个样点，这对本身存储器资源就比较紧张的低成本经济型方案中的 FPGA 来说，实现起来十分困难。

2 基于 CORDIC 算法的 NCO 实现原理

2.1 CORDIC 算法基本原理

坐标旋转数值计算(coordinate rotations digital computer，CORDIC)算法由 J. Volder 于 1959 年提出，首先用于导航系统，使得矢量的旋转和定向运算不需要做查三角函数表、乘法、开方及反三角函数等复杂运算[3-6]。

如图 2，假设 $(\boldsymbol{x}_0,\boldsymbol{y}_0)$ 为初始向量，$(\boldsymbol{x}_\mathrm{a},\boldsymbol{y}_\mathrm{a})$ 为 $(\boldsymbol{x}_0,\boldsymbol{y}_0)$ 经过旋转角度 θ 之后得到目标向量，则对于旋转前后的这 2 个向量有简单的如下关系：

$$\begin{aligned} x_\mathrm{a} &= \cos\theta(x_0 - y_0\tan\theta) \\ y_\mathrm{a} &= \cos\theta(y_0 - x_0\tan\theta) \end{aligned} \tag{1}$$

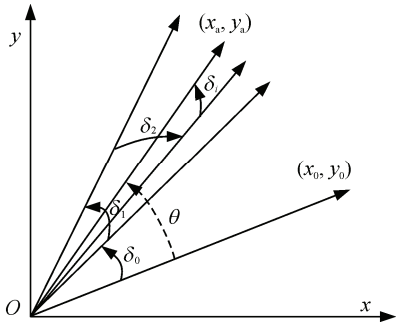


图 2 CORDIC 算法坐标旋转图示

假设初始向量是经过 n 次旋转逐步逼近得到的目标向量，旋转过程如图 2，并且假设每一次旋转的角度 $\delta_i = \arctan 2^{1-i}$，则有 $\cos\delta_i = \sqrt{\dfrac{1}{1+2^{2-2i}}}$。

引入变量 z_i 用以表示进行 i 次旋转后与目标角度之差，同时引入变量 $S_i = \{-1,1\}$ 用以表示旋转角度的方向[7]，容易得到角度 $\theta = \sum_{i=0}^{n} S_i\delta_i$。

第 $i+1$ 步旋转可以表示为：

$$\begin{aligned} x_{i+1} &= \sqrt{\frac{1}{1+2^{-2i}}}(x_i - S_i y_i 2^{-i}) \\ y_{i+1} &= \sqrt{\frac{1}{1+2^{-2i}}}(y_i + S_i x_i 2^{-i}) \\ z_{i+1} &= z_i - S_i\arctan(2^{-i}) \end{aligned} \tag{2}$$

其中，当 $z_i \geqslant 0$ 时，$S_i = 1$；当 $z_i \leqslant 0$ 时，$S_i = -1$。

在式 (2) 中可以先不考虑 $\sqrt{\dfrac{1}{1+2^{2-2i}}}$ 的影响，在完成 n 次旋转迭代之后再对计算结果进行修正。则该算法的每一步公式可以简化为：

$$\begin{aligned} x_{i+1} &= x_i - S_i y_i 2^{-i} \\ y_{i+1} &= y_i + S_i x_i 2^{-i} \\ z_{i+1} &= z_i - S_i\arctan(2^{1-i}) \end{aligned} \tag{3}$$

可以看出，前面的向量旋转在这里可以通过该算法方便地转换为硬件的加减法和移位操作。

初始向量 $(\boldsymbol{x}_0,\boldsymbol{y}_0)$，目标向量 $(\boldsymbol{x}_\mathrm{a},\boldsymbol{y}_\mathrm{a})$、$(\boldsymbol{x}_0,\boldsymbol{y}_0)$，经过 n 次旋转后所得向量 $(\boldsymbol{x}_n,\boldsymbol{y}_n)$ 之间有如下关系：

$$\begin{aligned} x_\mathrm{a} &= \cos\theta(x_0 - y_0\tan\theta) \approx Kx_n \\ y_\mathrm{a} &= \cos\theta(y_0 + x_0\tan\theta) \approx Ky_n \\ z_n &\to 0 \end{aligned} \tag{4}$$

其中 $K = \prod_{i=0}^{\infty}\sqrt{\dfrac{1}{1+2^{2-2i}}} \approx 0.607\ 252\ 935\ 01$ 称为模校因子[3-6]。

2.2 CORDIC 算法实现 NCO 的工作原理

基于 CORDIC 算法实现的 NCO 主要由相位累加器、相位加法器、相位处理单元、CORDIC 运算单元和输出数据处理单元 5 部分组成，其结构如图 3 所示。

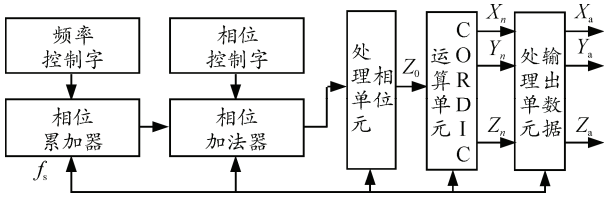


图 3 CORDIC 算法实现 NCO 的工作原理

由 $\delta_i = \arctan\ 2^{1-i}$ 可以看出，迭代的初始旋转角度最大为 45°，这样可以得出旋转角 θ 的范围是和迭代次数 N 相关的，关系如下式：

$$-\sum_{i=1}^{N}\arctan(2^{1-i}) \leqslant \theta \leqslant \sum_{i=0}^{N}\arctan(2^{1-i}) \tag{5}$$

容易得到 θ 的角度范围为[−99.883°, 99.883°]。为了能得到[0°, 360°]的角度旋转，在这里通过相位处理单元对输入 CORDIC 运算单元的相位进行折叠，使其落在[0°, 90°]的 1/4 圆范围内，然后 CORDIC 运算单元的输出经过输出数据处理单元，对输出数据进行处理。这样，输入任意相位角度 θ，都能得出其对应的 $\sin\theta$ 和 $\cos\theta$ 函数值。相位处理单元的角度折叠和输出数据处理单元的函数值对应处理关系如表 1。

表 1　相位折叠及函数值对应关系

原始相位最高两位	原始相位角度 ∂ 范围	折叠角 β 和 ∂ 之间的关系	∂ 和 β 函数值之间的关系
[00]	[0°, 90°)	$\beta=\partial$	$\sin\partial=\sin\beta,\cos\partial=\cos\beta$
[01]	[90°, 180°)	$\beta=\partial-90°$	$\sin\partial=\cos\beta,\cos\partial=-\sin\beta$
[10]	[180°, 270°)	$\beta=\partial-180°$	$\sin\partial=-\sin\beta,\cos\partial=-\cos\beta$
[11]	[270°, 360°)	$\beta=\partial-270°$	$\sin\partial=-\cos\beta,\cos\partial=\sin\beta$

为提高 CORDIC 算法的运算效率，在实现 NCO 时采用 CORDIC 算法的流水线结构[8-9]，如图 4。

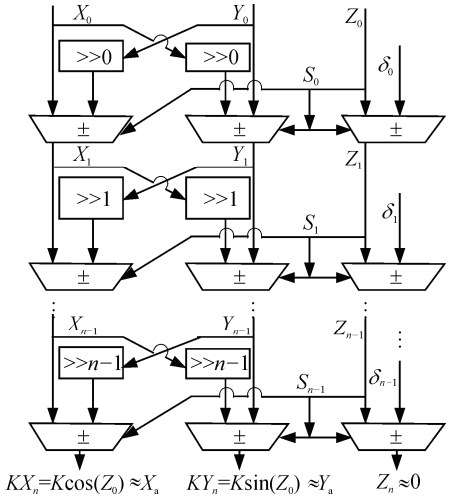


图 4　CORDIC 算法的流水线结构

由前面对 CORDIC 算法的介绍可以知道，该算法是通过一系列的迭代运算完成初始向量向目标向量的逼近，假设希望输出信号的相位角度误差小于 $\arctan(2^{-15})$，那么需要进行 16 次迭代运算。虽然计算精度会随着迭代级数的增加而增加，但是对于 FPGA 来说硬件资源消耗也会进一步增大，并且振荡器的调谐等待时间变长，功耗增大。

3　混合型的 NCO 实现原理

所谓混合型的 NCO，就是将查找表法和 CORDIC 算法相结合来实现的 NCO。其结构如图 5。

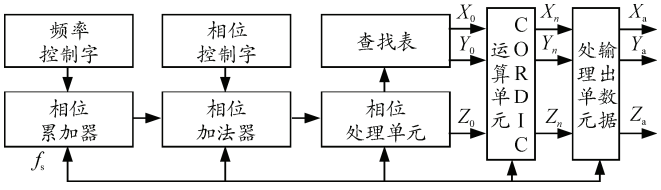


图 5　混合型 NCO 的工作原理

由图 3 和图 5 可以看出，混合型的 NCO 和基于 CORDIC 算法的 NCO 之间的区别在于：前者的 CORDIC 运算单元的输入（初值）是由查找表得来的，并且其初始旋转角度和模校因子也发生了变化。

3.1　CORDIC 运算单元初值的确定

如果将单位圆平均分割为 1 024 份，并将第一象限内的每个扇形区域的起始向量坐标值存储在查找表中，那么可以通过象限折叠在查找表中得出任意扇形区域的起始向量的坐标值。

假设频率控制字（FCW）、相位累加器、相位控制字（PCW）和相位加法器均为 32 位。经过相位加法器产生的初始相位中，高 2 位的 4 个不同值分别代表 4 个象限，30～23 位作为查找表地址，通过查找表得出目标向量所处于的小扇形区域的起始向量，低 22 位用来表示由小扇形区域的起始向量到目标向量的旋转角度 z_0。查找表得出的向量坐标 $(\boldsymbol{x}_0,\boldsymbol{y}_0)$ 和旋转角度 z_0 作为 CORDIC 运算单元的初值。这样，只需经过 8 级迭代运算就能使输出信号的相位角度误差小于 $\arctan(2^{-15})$。

3.2　初始旋转角和迭代级数的确定

由 3.1 节可知每个小扇形区域的弧度大小为 $\sigma=\dfrac{2\pi}{1\,024}\approx 0.006\,135\,9$。假设该 CORDIC 运算单元的第一次旋转相当于单独运用 CORDIC 算法进行的第 i 次旋转，为了保证输出信号的相位角精度并且最大程度地减小旋转的次数，则第 i 次旋转的角度需要满足 $\delta_i\leqslant\sigma\leqslant\delta_{i-1}$。由表 2 可知 $\delta_9<\sigma<\delta_8$，$i=9$。所以，这里的 CORDIC 运算单元的第一次旋转的角度（弧度表示）$\alpha_1=0.003\,906\,230\,132$。因此，如果希望输出信号的相位角度误差小于 $\arctan(2^{-15})$，那么还需要进行 8 次迭代运算。

表 2　CORDIC 旋转级数及旋转角度对应关系

旋转级数 i	$\tan\delta_i$	δ_i/rad	$\cos\delta_i$
1	1.000 000 000 000	0.785 398 163 397	0.707 106 781 186
2	0.500 000 000 000	0.463 647 609 001	0.894 427 190 999
3	0.250 000 000 000	0.244 978 663 126	0.970 142 500 145
4	0.125 000 000 000	0.124 354 994 546	0.992 277 876 713
5	0.062 500 000 000	0.062 418 809 995	0.998 052 578 482
6	0.031 250 000 000	0.031 239 833 430	0.999 512 076 087
7	0.015 625 000 000	0.015 623 728 620	0.999 877 952 034
8	0.007 812 500 000	0.007 812 341 060	0.999 969 483 818
9	0.003 906 250 000	0.003 906 230 131	0.999 992 370 692
10	0.001 953 125 000	0.001 953 122 516	0.999 998 092 656
11	0.000 976 562 500	0.000 976 562 189	0.999 999 523 163
12	0.000 488 281 250	0.000 488 281 211	0.999 999 880 790
13	0.000 244 140 625	0.000 244 140 620	0.999 999 970 197
14	0.000 122 070 312	0.000 122 070 311	0.999 999 992 549
15	0.000 061 035 157	0.000 061 035 156	0.999 999 998 137
16	0.000 030 517 578	0.000 030 517 578	0.999 999 999 534

3.3　模校因子的确定

由于采用了查找表，初值和初始旋转角都发生了变化，因此模校因子也需要做相应的调整。

在这里，初始旋转角度为 $\alpha_1=\delta_9$，因此新的模校

因子为：

$$K=\prod_{i=9}^{\infty}\sqrt{\frac{1}{1+2^{2-2i}}}\approx 0.999\ 989\ 827\ 59 \tag{6}$$

4 仿真与结果分析

按照图 1、图 3、图 5 所示的 NCO 的 3 种不同实现方式，用 Verilog HDL 语言编程分别实现这 3 种 NCO，在集成开发环境 ISE13.4 中以 XCV5LX155T 为平台进行分析综合、布局布线，并通过调用 Modelsim6.4SE 进行仿真。设频率控制字为 02 852 F13h，相位控制字为 00 000 000h。其中混合型 NCO 的仿真结果如图 6。

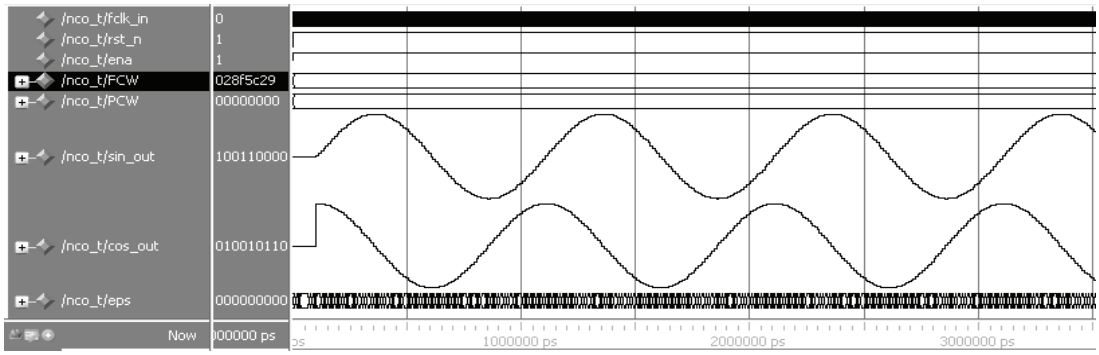
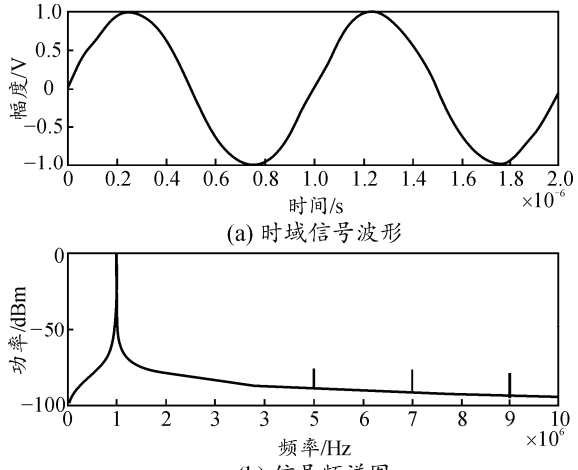


图 6 混合型 NCO 的时序仿真结果

将仿真产生的波形信号导出，并在 Matlab 中进行波形恢复，其时域和频域的波形如图 7。



(a) 时域信号波形

(b) 信号频谱图

图 7 Matlab 波形恢复及频谱分析

3 种不同实现方式实现的 NCO 经过分析综合后所消耗的硬件资源情况及延时对比如表 3 所示。

表 3 3 种不同方式实现的 NCO 消耗资源情况

硬件资源类别	查找表法	CORDIC 算法	混合型
Slice Registers	112	1 589	926
Slice LUT	172	1 609	860
Memory/kb	1 044	0	36
延时(时钟周期)	3	18	10

由表 3 可见，在实现相同精度的信号输出时，混合型 NCO 通过 256 点深度的查找表和 8 级流水迭代旋转相结合，以消耗少量的存储器资源为代价节省了将近一半的逻辑资源，并且该混合型 NCO 的延时也只有 10 个时钟周期。

5 结束语

仿真结果表明：用查找表和 CORDIC 算法相结合的方法实现的 NCO 能很好地平衡 FPGA 上存储器资源和逻辑资源的消耗，并且在产生信号的精度、速度、纯度方面也有很好的兼顾。另外，混合型 NCO 还可以根据所选器件的硬件资源，通过改变查找表的深度和 CORDIC 迭代运算次数来改变这 2 种硬件资源的消耗，具有较高的实用价值。

参考文献：

[1] 王玉良, 李宏生, 夏敦柱. 基于CORDIC算法的NCO在FPGA中的实现[J]. 计算机与数字工程, 2009, 242(12): 21-23.

[2] 侯宏录, 姚恩源. 基于FPGA的La Gell 5/3小波变换的硬件电路设计[J]. 兵工自动化, 2013, 32(2): 59-62.

[3] 王春来. 数字下变频的 FPGA 实现[D]. 郑州: 解放军信息工程大学, 2009: 4-7.

[4] 杨勋. 软件无线电中上下变频技术的设计和实现[D]. 西安: 西安电子科技大学, 2007: 23-34.

[5] 卢笛. 基于 FPGA 的多速率数字信号上下变频的实现[D]. 西安: 西安电子科技大学, 2012: 24-27, 49-52.

[6] Uwe Meyer. Baese. 数字信号处理的 FPGA 实现[M]. 刘凌, 等, 译. 北京: 清华大学出版社, 2006: 168-182.

[7] 李荣, 李响, 申志伟. 向量合成加速单纯形算法[J]. 四川兵工学报, 2013, 34(8): 138-140.

[8] 何宾. FPGA 数字信号处理实现原理及方法[M]. 北京: 清华大学出版社, 2010: 105-121.

[9] Andraka R. A survey of CORDIC algorithms for FPGA based computers[C]. North Kingstown: Proceedings of the 1998 ACM/SIGDA 6th international symposium on FPGA, 1998: 191-200.