文章编号：1674-2974（2023）12-0069-07 DOI：10.16339/j.cnki.hdxbzkb.2023299

高精度低消耗CORDIC算法设计

姚亚峰[1]，杨金岷[1]，周群群[1†]，付东兵[2]
（1. 中国地质大学 机械与电子信息学院，湖北 武汉 430074；
2. 重庆吉芯科技有限公司，重庆 400060）

摘 要：针对CORDIC算法存在硬件资源消耗大、输出精度低等问题，提出一种基于区间合并迭代的改进CORDIC算法.算法在两段式CORDIC算法的基础上，采用区间合并迭代来完成第二阶段的合并迭代运算.针对合并迭代中移位运算产生的截位误差，区间合并迭代通过减少数据移位的大小和次数来减少在合并迭代过程中产生的数据误差和资源消耗. 仿真结果表明，改进CORDIC算法不仅保留了两段式算法在低时延上的良好特性，在寄存器消耗上也相比基本算法减少36.8%，相比三段式和两段式算法分别减少14.8%和9.5%.当给定16 bit的输出位宽时，改进算法的平均误差相比基本算法降低37.0%，相比三段式和两段式算法分别降低19.4%和24.5%，因此更适用于高速、高精度、低消耗的现代数字通信.

关键词：坐标旋转计算机；角度二极化重编码；区间合并迭代；数字信号处理
中图分类号：TN492 **文献标志码**：A

Design of CORDIC Algorithm with High Accuracy and Low Consumption

YAO Yafeng[1]，YANG Jinmin[1]，ZHOU Qunqun[1†]，FU Dongbing[2]
（1. School of Mechanical Engineering and Electronic Information，China University of Geosciences，Wuhan 430074，China；
2. Chongqing Jixin Technology Co.，Ltd.，Chongqing 400060，China）

Abstract：Aiming at the problems of high hardware resource consumption and low output accuracy in CORDIC algorithms，this design proposes an improved Coordinate Rotation Digital Computer（CORDIC）algorithm based on interval merging iteration. Based on the two-stage CORDIC algorithm，the algorithm uses interval merging iteration to complete the merging iteration operation in the second stage. For the truncation error caused by shift operation in merge iteration，interval merge iteration reduces the data error and resource consumption generated during the merge iteration process by reducing the size and number of data shifts. The simulation results show that the improved CORDIC algorithm not only retains the good characteristics of the two-stage algorithm in low latency but also reduces register consumption by 36.8% compared with the basic algorithm，14.8% and 9.5% compared with the three-segment and two-stage algorithms，respectively. When a 16 bit output bit-width is given，the average error of the improved algorithm is reduced by 37.0% compared with the basic algorithm，19.4% and 24.5% respectively compared with the three segment and two segment algorithms. Therefore，it is more suitable for modern digital communication with high speed，high accuracy，and low consumption.

* **收稿日期**：2023-02-27
基金项目：模拟集成电路国家重点实验室稳定支持项目（JCKY2019210C058），Stability Support Project of State Key Laboratory of Analog（JCKY2019210C058）
作者简介：姚亚峰（1970—），男，湖北黄梅人，中国地质大学（武汉）副教授，博士
† 通信联系人，E-mail：787458282@qq.com

Key words: coordinate rotation digital computer(CORDIC); angle bipolar recoding; interval merging iteration; digital signal processing

坐标旋转计算机(Coordinate Rotation Digital Computer, CORDIC)于1945年由Volder提出,它是通过将坐标进行旋转,并利用不断旋转的角度进行叠加,最终逼近初始角度,从而求取函数值.该算法能够用统一的加减、移位操作来代替多种难以用硬件电路直接实现的复杂函数运算,降低了硬件设计的复杂性,同时在速度和精度方面也有显著提高,被广泛用于直接数字频率合成器(Direct Digital Synthesizer, DDS)、数字控制振荡器(Numerically Controlled Oscillator, NCO)、快速傅里叶变换(Fast Fourier Transform, FFT)[1-3]等领域中.在三角函数运算、矩阵相关运算等多种数字信号处理中[4-6]也都得到广泛应用.随着集成电路技术的不断发展,以及超大规模集成电路VLSI的产生,总是需要用硬件设备实现超高速和超高精度的函数计算,因此针对CORDIC算法的研究从未间断,CORDIC算法及其应用研究仍然是数字通信和信号处理领域研究热点之一.文献[7]基于一种高效的非迭代NR-8实现CORDIC算法,仅使用[0, π/12]的窄范围内的角度而不是[0, π/2]的宽角度范围内的角度计算函数值,输出时延以及资源消耗相对基本CORDIC算法都有所减少,但是输出数据精度仅为0.012%.文献[8]基于并行流水线混合自适应架构的设计实现了低功耗的CORDIC算法,但相比于基本CORDIC算法在输出时延和精度方面并没有得到改善.文献[9-10]根据欠阻尼理论,将固定角度旋转采用单向旋转CORDIC算法实现,减少了流水线的级数和迭代符号位的判决,虽然相对于基本CORDIC算法减少了迭代次数,并且资源消耗较低,但其输出数据精度也仅在10^{-4}数量级.文献[11]通过角度预处理来扩大收敛区间,从而采用双向同步旋转的迭代方式,提升了CORDIC算法的精度和健壮性,但是算法仍存在较大的输出时延.文献[12]提出了一种基于基-2时间抽取算法的改进多径延迟换向器流水线结构的改进算法,具有更少的输出延迟和更高的硬件利用率,但是在精度上相比其他改进算法较低.文献[13-14]提出改进将CORDIC算法实现划分为三个阶段实现,第一阶段通过建立查找表来一步实现前几次迭代,第二阶段将中间迭代过程通过移位-加法蝶形运算来代替,第三阶段通过合并迭代来完成,该改进算法相比于传统CORDIC算法无论是在资源消耗还是输出精度上都有所提高,但仍然存在较大的输出时延.文献[15]进一步在三段式改进CORDIC算法的基础上提出了两段式的改进CORDIC算法,省去了三段式CORDIC算法中间蝶形迭代的步骤,仅通过建立查找表和合并迭代两部进行实现,解决了三段式输出时延较大的问题,但在资源消耗和输出精度上仍然可以进一步改进.

本文基于角度二极化重编码、区间合并迭代、截位误差估计等技术在两段式算法的基础上进行改进,通过对截位误差的分析,将第二阶段的合并迭代采用区间合并迭代的方式替代,将多个较大的移位操作进行区间合并,减少数据移位的大小和次数,从而降低在合并迭代过程中产生的数据误差和资源消耗,相比于传统算法和两段式算法,该改进算法不仅保持了两段式算法在输出时延上面的良好性能,还显著减少了资源的消耗,在输出数据的精度和功耗上也有着明显的提升.

1 CORDIC算法基本实现原理

CORDIC算法是一种数值近似法,它通过一系列的角度旋转迭代不断逼近计算的角度,得到相应的正余弦值.图1为CORDIC算法的平面旋转图.

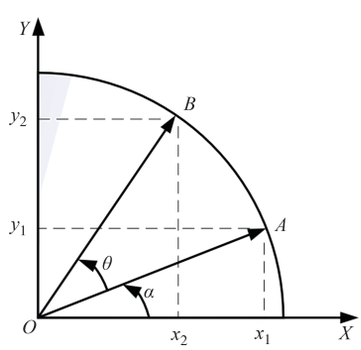


图1 平面旋转图

Fig.1 Plan rotation diagram

在该平面旋转图中从A点旋转角度θ至B点,坐标关系可表示为公式(1):

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

公式(1)是进行一次角度旋转得到的表达式,将θ角细分为若干个小角度,即$\theta=\sum_{i=0}^{N}\theta_i$,分步旋转整个$\theta$角可以得到公式(2):

$$\begin{cases} x_\theta = K\prod_{i=0}^{N}(x_i - y_i\tan\theta_i) \\ y_\theta = K\prod_{i=0}^{N}(y_i + x_i\tan\theta_i) \end{cases} \tag{2}$$

公式(2)中$K=\prod_{i=0}^{N}\cos\theta_i$,可以看出CORDIC算法的核心在于$\theta_i$的取值,通过选取一些特定的值,不仅可以让$K$值固定,同时将公式(2)中复杂的乘法运算转化为简单的移位运算,大大减少了计算量和复杂度,因此选取$\tan\theta_i=2^{-i}$,选取特定的一组θ_i之后,每次旋转按照固定的角度进行旋转,引入变量z表示每次旋转之后的剩余角度,即公式(3):

$$\begin{cases} z_{N+1} = \theta - \sum_{i=0}^{N} d_i\theta_i \\ d_i = \begin{cases} 1 & (z_N > 0) \\ -1 & (z_N < 0) \end{cases} \end{cases} \tag{3}$$

其中d_i为角度判断算子,当$d_i=1$时进行逆时针旋转,当$d_i=-1$时进行顺时针旋转,按此旋转则可以让$z_{N+1}\to 0$,由此化简公式(2)得最终旋转公式(4):

$$\begin{cases} x_\theta = K\prod_{i=0}^{N}(x_i - y_i d_i 2^{-i}) \\ y_\theta = K\prod_{i=0}^{N}(y_i + x_i d_i 2^{-i}) \end{cases} \tag{4}$$

当迭代次数N值确定时,K的取值也随之确定,通常将初始向量取为$(K,0)$,经过一系列的角度θ_i旋转来逼近所要计算的角度值,最后得到另一向量$(x_\theta,\ y_\theta)=(\cos_\theta,\ \sin_\theta)$,该坐标值即为所求角度的正余弦值.

2　改进CORDIC算法

2.1　角度区间折叠及映射关系

传统基本算法通过旋转,迭代计算$[\pi/2:0]$区间内任意角度的值,其他区间内的角度则通过角度区间折叠和映射的数学关系进行计算,无须通过旋转,迭代来进行计算,改进算法将初始旋转区间从$[\pi/2:0]$压缩至$[\pi/8:0]$,此区间之外的任意角度通过角度区间折叠和映射的数学关系来进行计算,具体角度映射转化关系如表1所示.

表1　角度映射关系

Tab.1　Angle mapping

输入角度θ	预处理后的角度θ'	余弦值$\cos\theta$	正弦值$\sin\theta$
$[0,\pi/8]$	θ	$\cos\theta'$	$\sin\theta'$
$(\pi/8,\pi/4]$	$\pi/4-\theta$	$\frac{\sqrt{2}}{2}(\cos\theta'+\sin\theta')$	$\frac{\sqrt{2}}{2}(\cos\theta'-\sin\theta')$
$(\pi/4,3\pi/8]$	$\theta-\pi/4$	$\frac{\sqrt{2}}{2}(\cos\theta'-\sin\theta')$	$\frac{\sqrt{2}}{2}(\cos\theta'+\sin\theta')$
$(3\pi/8,\pi/2]$	$\pi/2-\theta$	$\sin\theta'$	$\cos\theta'$
$(\pi/2,5\pi/8]$	$\theta-\pi/2$	$-\sin\theta'$	$\cos\theta'$
$(5\pi/8,3\pi/4]$	$3\pi/4-\theta$	$\frac{\sqrt{2}}{2}(\cos\theta'-\sin\theta')$	$\frac{\sqrt{2}}{2}(\cos\theta'+\sin\theta')$
$(3\pi/4,7\pi/8]$	$\theta-3\pi/4$	$\frac{\sqrt{2}}{2}(\cos\theta'+\sin\theta')$	$\frac{\sqrt{2}}{2}(\cos\theta'-\sin\theta')$
$(7\pi/8,\pi]$	$\pi-\theta$	$-\cos\theta'$	$\sin\theta'$
$(\pi,9\pi/8]$	$\theta-\pi$	$-\cos\theta'$	$-\sin\theta'$
$(9\pi/8,5\pi/4]$	$5\pi/4-\theta$	$\frac{\sqrt{2}}{2}(\cos\theta'+\sin\theta')$	$\frac{\sqrt{2}}{2}(\cos\theta'-\sin\theta')$
$(5\pi/4,11\pi/8]$	$\theta-5\pi/4$	$\frac{\sqrt{2}}{2}(\cos\theta'-\sin\theta')$	$\frac{\sqrt{2}}{2}(\cos\theta'+\sin\theta')$
$(11\pi/8,3\pi/2]$	$3\pi/2-\theta$	$-\sin\theta'$	$-\cos\theta'$
$(3\pi/2,13\pi/8]$	$\theta-3\pi/2$	$\sin\theta'$	$-\cos\theta'$
$(13\pi/8,7\pi/4]$	$7\pi/4-\theta$	$\frac{\sqrt{2}}{2}(\cos\theta'-\sin\theta')$	$\frac{\sqrt{2}}{2}(\cos\theta'+\sin\theta')$
$(7\pi/4,15\pi/8]$	$\theta-7\pi/4$	$\frac{\sqrt{2}}{2}(\cos\theta'+\sin\theta')$	$\frac{\sqrt{2}}{2}(\cos\theta'-\sin\theta')$
$(15\pi/8,2\pi]$	$2\pi-\theta$	$\cos\theta'$	$-\sin\theta'$

2.2 角度二极化重编码和建立查找表

根据角度重编码原理,给定一个任意小于1 rad的正角度θ,它可以表示为$\theta=\sum_{i=1}^{N}b_i\theta_i$,其中$b_i\in\{0,1\}$,$\theta_i=2^{-i}$,$\theta$包含一位符号位和$N$位二进制小数位,因为角度区间被进一步缩小至$[\pi/8:0]$,符号位不参与$\theta$角的求和.二进制$b_i\in\{0,1\}$可以被编码到有符号的数字中去,即用$b_i$的值来表示$r_i=\{1,-1\}$,如公式(5)所示:

$$\begin{cases}\theta=\sum_{i=2}^{N}b_i2^{-i}=\varphi_0+\sum_{i=3}^{N+1}r_i2^{-(i+1)}\\ \varphi_0=\sum_{i=3}^{N+1}2^{-i}=\frac{1}{4}-\left(\frac{1}{2}\right)^{N+1}\\ r_i=2b_{i-1}-1\end{cases}\tag{5}$$

不同于基本CORDIC算法每次旋转角度$\arctan(2^{-i})$ rad,公式(5)计算θ的值分为两部分,第一部分先旋转角度φ_0,再根据r_i的值依次进行2^{-i} rad的方向旋转,同时为省去传统CORDIC算法单次迭代中对旋转方向的计算,从而节省硬件资源消耗,只是在理论上进行$b_i\in\{0,1\}$到$r_i+1\in\{-1,1\}$的映射,使每次迭代方向由输入角度二进制的位值直接确定,避免了传统算法中迭代方向需要每次计算旋转之后剩余角度的大小来确定,从而降低了算法的资源消耗.

在进行角度重编码后可建立相应的查找表,改进算法将角度重编码范围压缩至$[\pi/8:0]$,旋转初始值i则是从$i=3$开始,所以给定一个B bit的数据位宽,当$3\leqslant i<(B-1)/2$建立相应的查找表,将进行角度φ_0以及前$[(B-1)/2-3]$次旋转后的结果乘以k_n缩放得到的值存储在查找表中,k_n以及初始角度旋转坐标计算如公式(6)所示,其中$(x_0,y_0)=(k_n,0)$.

$$\begin{cases}k_n=\cos(\varphi_0)\prod_{i=3}^{N+1}\cos(2^{-i})\\ x_3=x_0-\tan(\varphi_0)\cdot y_0\\ y_3=y_0+\tan(\varphi_0)\cdot x_0\end{cases}\tag{6}$$

2.3 区间合并迭代

将传统CORDIC算法迭代公式(4)展开进行两步迭代得到公式(7):

$$\begin{cases}x_{i+2}=x_i-y_ir_i2^{-i}-x_ir_ir_{i+1}2^{-2i-1}-\\ \qquad y_ir_{i+1}2^{-i-1}\\ y_{i+2}=y+x_ir_i2^{-i}-y_ir_ir_{i+1}2^{-2i-1}+\\ \qquad x_ir_{i+1}2^{-i-1}\end{cases}\tag{7}$$

给定一个B bit的数据位宽,当迭代次数$i\geqslant(B-1)/2$时,上式中的$x_ir_ir_{i+1}2^{-2i-1}$和$y_ir_ir_{i+1}2^{-2i-1}$近似为0,所以当$i\geqslant(B-1)/2$时,迭代公式进一步化简为公式(8):

$$\begin{cases}x_{i+2}=x_i-y_ir_i2^{-i}-y_ir_{i+1}2^{-i-1}\\ y_{i+2}=y_i+x_ir_i2^{-i}+x_ir_{i+1}2^{-i-1}\end{cases}\tag{8}$$

再结合二进制移位计算的原理,可以只通过加减和移位操作,得到任意旋转的x_i和y_i,即当$i\geqslant(B-1)/2$时的最终合并迭代为公式(9):

$$\begin{cases}x_{i+m}=x_i-y_i\sum_{k=i}^{i+m-1}r_k2^{-k}\\ y_{i+m}=y_i+x_i\sum_{k=i}^{i+m-1}r_k2^{-k}\end{cases}\tag{9}$$

公式(9)中2^{-k}在Verilog HDL语言中通过移位操作来实现,但是在移位操作之后,为了减少资源的消耗,移位扩宽后的数据位宽会采取截位的操作,则会产生一定的误差,因此在进行移位-截位操作之前,当$k\geqslant(B-1)/2$时,采用区间合并的方式,通过对截位误差的估计采取最佳的移位-截位方式,通过将多次多位的移位合并成一个区间来进行更少次数更少位数的移位来减小截位误差的产生,区间合并移位如图2和表2所示.

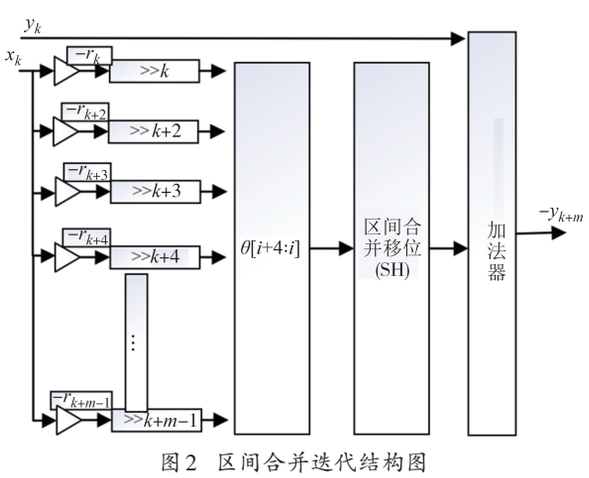


图2 区间合并迭代结构图

Fig.2 Interval merging iteration structure chart

2.4 改进CORDIC算法整体框图

根据上述原理描述,改进CORDIC算法主要包含查找表、区间合并迭代、累加求和、角度区间映射几个阶段,查找表通过MATLAB建立,对整个算法通过Verilog HDL语言进行编程实现,设计输入角度位宽为20 bit(其中高四位为角度区间范围),设计输出位宽为16 bit(1 bit符号位+15 bit数据位),算法的整体设计框图如图3所示.

表2　区间合并迭代计算表

Tab.2　Interval consolidation iteration calculation table

角度/(θ)	角度二进制码	区间合并移位/SH
θ[i + 4: i]	4'b0 000	>>k−1−>>k+3
	4'b0 001	>>k−1−>>k+2+>>k+3
	4'b0 010	>>k−1−>>k+2−>>k+3
	4'b0 011	>>k+>>k+3
	4'b0 100	>>k−>>k+3
	4'b0 101	>>k+1+>>k+3
	4'b0 110	>>k+1−>>k+3
	4'b0 111	>>k+3
	4'b1 000	>>k+3
	4'b1 001	>>k+1−>>k+3
	4'b1 010	>>k+1+>>k+3
	4'b1 011	>>k−>>k+3
	4'b1 100	>>k+>>k+3
	4'b1 101	>>k−1−>>k+2−>>k+3
	4'b1 110	>>k−1−>>k+2+>>k+3
	4'b1 111	>>k−1−>>k+3

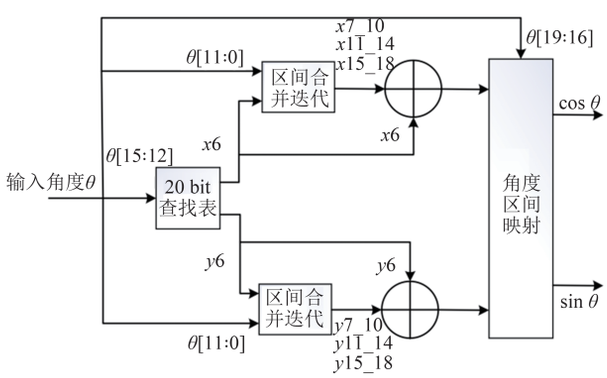


图3　算法整体设计框图

Fig.3　Overall design block diagram of algorithm

3　仿真结果及其分析

将改进高精度低消耗CORDIC算法,三段式CORDIC算法,两段式CORDIC算法以及基本CORDIC算法分别通过Verilog HDL语言进行编程实现,选用XILINX公司型号为xc7z020iclg400−1L的FPGA,在VIVADO软件环境下进行功能仿真,同时利用其自带综合工具对四种实现方式进行综合,通过综合结果对硬件资源消耗、功耗和输出时延进行分析.

表3给出了四种算法资源消耗和输出时延对比结果,改进算法在寄存器消耗上相比基本算法减少了36.8%,相比三段式和两段式算法分别减少了14.8%和9.5%,LUT的消耗数量相比基本算法减少了41.1%,与两段式算法基本一致. 改进算法的最大输出时延与两段式算法保持一致,仅需要2个时钟周期,相比于基本算法16个时钟周期和三段式算法7个时钟周期有着很大的改善.总的来说改进CORDIC算法在资源消耗和输出时延上相比其他三种算法都有大幅度的改善,总体性能有了较大的提升.

表4给出了四种算法随着时钟频率的增大功耗对比的结果,当时钟运行频率较低时,改进算法和两段式算法功耗消耗一致,但相对于基本算法有着较大的提升,随着时钟频率的增加,改进算法的功耗相对于两段式算法也有一定的改善.

表3　综合结果对比

Tab.3　Comparison of comprehensive results

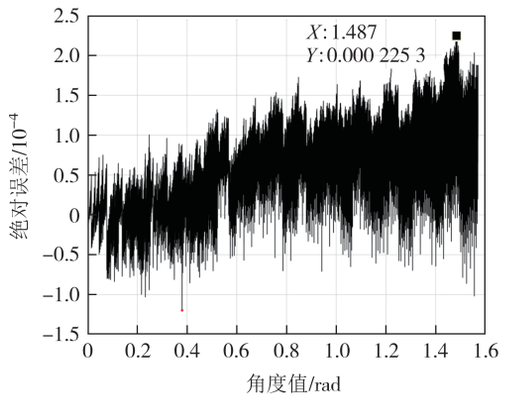
CORDIC算法类型	寄存器消耗数量/个	LUT消耗数量/个	最大输出时延/时钟周期
基本	855	1 879	16
三段式	634	981	7
两段式	597	1 088	2
改进	540	1 107	2

表4　功耗对比表

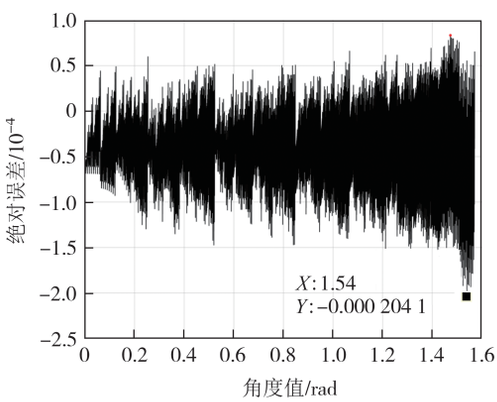
Tab.4　Power consumption comparison table

CORDIC算法类型	运行频率/MHz			
	30	60	90	125
基本	0.114	0.139	0.164	0.174
三段式	0.104	0.120	0.136	0.154
两段式	0.106	0.124	0.142	0.163
改进	0.106	0.124	0.141	0.161

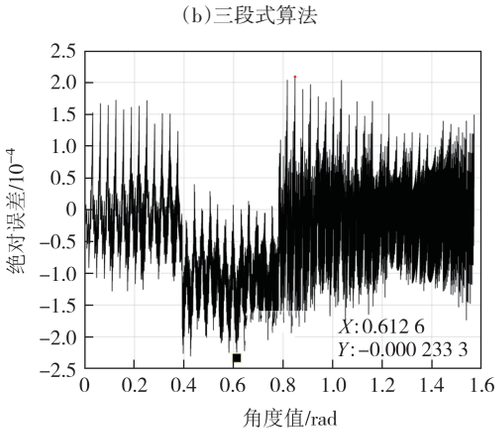
通过MTLAB对四种算法进行建模,分析各自的输出精度,以2^{-16} rad分辨率遍历[0: π/2]得到四种算法产生的余弦值,再与以相同角度分辨率产生的标准余弦函数作差得到相应的误差值,将所有误差值取绝对值后,进行算术平均得到平均误差.图4给出了输出位宽为16 bit时四种算法的误差仿真结果,基本算法、三段式算法、两段式算法和改进算法四者余弦值最大误差分别为0.000 225 3,0.000 204 1,0.000 233 3,0.000 205 1,此时改进算法的平均误差相对于基本算法降低了37.0%,相对于三段式两段式算法分别降低了19.4%和24.5%.表5给出了不同输出位宽下四种算法各自的平均误差,可以看出四种算法中改进算法的输出精度最优,特别是在低位宽输出时明显优于其他三种算法.



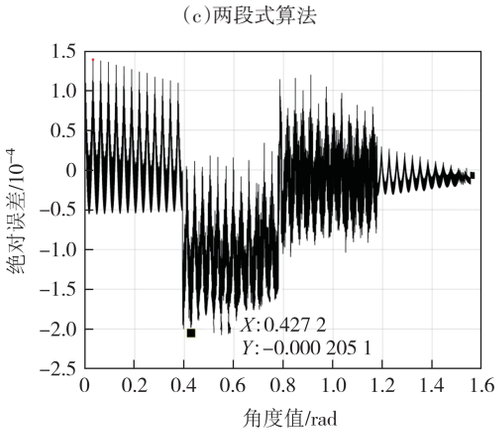
(a)基本算法



(b)三段式算法



(c)两段式算法



(d)改进算法

图4 输出位宽为16 bit绝对误差图

Fig.4 Absolute error diagram with output bit width of 16 bit

表5 输出精度对比表

Tab.5 Output precision comparison table

CORDIC算法类型	输出位宽/bit			
	12	14	16	18
基本	4.30×10^{-4}	1.34×10^{-4}	6.81×10^{-5}	6.88×10^{-5}
三段式	6.23×10^{-4}	1.63×10^{-4}	5.32×10^{-5}	2.44×10^{-5}
两段式	6.44×10^{-4}	1.97×10^{-4}	5.68×10^{-5}	2.51×10^{-5}
改进	3.40×10^{-4}	1.26×10^{-4}	4.29×10^{-5}	2.34×10^{-5}

4 结 论

本文阐述了一种基于区间合并迭代的改进CORDIC算法的实现原理,该算法在两段式CORDIC算法的基础上采用区间合并迭代的方式完成角度区间的合并,通过移位误差估计来免除不必要的移位,从而降低实现电路的资源消耗和功耗,并提升算法的输出精度.RTL代码的仿真结果表明本文提出的改进CORDIC算法保留了两段式算法在输出时延上的良好特性.通过MATLAB建模完成了四种算法的输出精度分析,并在XILINK公司型号为xc7z020iclg400-1L的FPGA开发版上完成四种算法的电路实现,使用VIVADO自带综合器进行综合.仿真结果表明:改进算法在寄存器消耗上相比基本算法减少了36.8%,相比三段式和两段式算法分别减少了14.8%和9.5%,LUT的消耗数量相比基本算法减少了41.1%,与两段式算法基本一致.在功耗分析上,当时钟运行频率较低时,改进算法和两段式算法功耗消耗一致,但相对于基本算法有着较大的提升,随着时钟频率的增加,改进算法的功耗相对于两段式算法也有着一定的改善.当输出位宽为16 bit时,改进算法的平均误差相对于基本算法降低了37.0%,相对于三段式和两段式算法分别降低了19.4%和24.5%.

目前该改进算法局限于以计算角度的四个二进制码为一组的区间合并,在这个范围内进行区间合并迭代能够在精度和资源消耗上都有较大的改善.当合并区间大于4时,对于精度可以进一步提高,但是资源消耗则会过大,这将是该改进算法未来的研究方向.

参考文献

[1] KUMAR A, KUMAR A, SINGH TOMAR G. Hardware chip performance of CORDIC based OFDM transceiver for wireless

communication[J]. Computer Systems Science and Engineering, 2022, 40(2): 645-659.

[2] ZHAO Y P, LV H, LI J, et al. High performance and resource efficient FFT processor based on CORDIC algorithm [J]. EURASIP Journal on Advances in Signal Processing, 2022, 2022(1):1-14.

[3] CHANGELA A, ZAVERI M, VERMA D. FPGA implementation of high-performance, resource-efficient Radix-16 CORDIC rotator based FFT algorithm[J]. Integration, 2020, 73: 89-100.

[4] FU W J, XIA J C, LIN X, et al. Low-latency hardware implementation of high-precision hyperbolic functions sinhx and coshx based on improved CORDIC algorithm[J]. Electronics, 2021, 10(20): 2533.

[5] PRATIK T, TANISH Z. Enhanced CORDIC based rotator design for sinusoidal transforms[J]. International Journal of Engineering and Advanced Technology, 2020, 9(3): 1001-1004.

[6] MOKHTAR A S N, KARIM S A, CHEW S P, et al. FPGA implementation of CORDIC algorithm in digital modulation[J]. Journal of Fundamental and Applied Sciences, 2018, 9(3S):279.

[7] TANG W M, XU F. A noniterative radix-8 CORDIC algorithm with low latency and high efficiency[J]. Electronics, 2020, 9(9): 1521.

[8] NGUYEN H T, NGUYEN X T, PHAM C K. A low-latency parallel pipeline CORDIC[J]. IEICE Transactions on Electronics, 2017, E100, C(4):391-398.

[9] 张朝柱, 韩吉南, 燕慧智. 高速高精度固定角度旋转CORDIC算法的设计与实现[J]. 电子学报, 2016, 44(2): 485-490.
ZHANG C Z, HAN J N, YAN H Z. Design and implementation of CORDIC algorithm for high speed and precision fixed angle of rotation[J]. Acta Electronica Sinica, 2016, 44(2): 485-490. (in Chinese)

[10] FANG L L, LI B Y, XIE Y Z, et al. A unified reconfigurable CORDIC processor for floating-point arithmetic[J]. International Journal of Electronics, 2020, 107(9): 1436-1450.

[11] 郑传喜, 古元冬. 高精度双向同步旋转CORDIC算法设计与实现[J]. 上海大学学报(自然科学版), 2022, 28(5): 872-882.
ZHENG C X, GU Y D. Design and implementation of a high-precision bidirectional synchronous rotation CORDIC algorithm [J]. Journal of Shanghai University (Natural Science Edition), 2022, 28(5): 872-882. (in Chinese)

[12] NGUYEN H N, KHAN A S, KIM C, et al. A high-performance, resource-efficient, reconfigurable parallel-pipelined FFT processor for FPGA platforms [J]. Microprocessors and Microsystems, 2018, 60: 96-106.

[13] NGUYEN H T, NGUYEN X T, PHAM C K. A low-power hybrid adaptive CORDIC[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2018, 65(4): 496-500.

[14] 侯强, 彭玉龙, 王育新, 等. 低时延CORDIC算法计算平方根电路设计研究[J]. 湖南大学学报(自然科学版), 2022, 49(2): 111-116.
HOU Q, PENG Y L, WANG Y X, et al. Study on design of low-delay CORDIC algorithm to calculate square-root circuit [J]. Journal of Hunan University (Natural Sciences), 2022, 49(2): 111-116. (in Chinese)

[15] YAO Y F, FENG Z X. BBR-based iteration-free CORDIC algorithm[J]. Journal of Circuits, Systems and Computers, 2018, 27(5): 1850076.