文章编号：1674-2974（2022）02-0111-06 DOI：10.16339/j.cnki.hdxbzkb.2022266

低时延CORDIC算法计算平方根电路设计研究

侯强[1]，彭玉龙[1]，王育新[2†]，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. 中国电子科技集团公司第24研究所 模拟集成电路国家重点实验室，重庆 400060）

摘 要：开平方运算广泛应用于数值分析、调制解调、图像处理等领域，而应用坐标旋转数字计算（Coordinate Rotation Digital Computer，CORDIC）进行平方根运算是一种新应用. 基本CORDIC算法精度必须用迭代次数作保证，而较多的迭代次数会导致时延过大等问题，通过运用建立查找表、单向旋转、合并迭代和免除补偿因子等手段，提出一种能够免去大部分迭代运算的改进CORDIC算法用于平方根计算. 相较于基本算法计算平方根，该改进算法使用了一半的时钟周期便能得到输出结果，大大减少了输出时延，而且可以达到较高的计算精度，更加适合实时性要求高的应用场合.

关键词：坐标旋转数字计算；平方根计算；单向旋转；合并迭代；数字计算机
中图分类号：TN492 **文献标志码**：A

Study on Design of Low-delay CORDIC Algorithm to Calculate Square-root Circuit

HOU Qiang[1]，PENG Yulong[1]，WANG Yuxin[2†]，FU Dongbing[2]
（1. School of Mechanical Engineering and Electronic Information，China University of Geosciences，Wuhan 430074，China；
2. State Key Laboratory of Analog Integrated Circuit，24th Research Institute，
China Electronics Technology Corporation，Chongqing 400060，China）

Abstract：Square root calculating is widely used in numerical analysis，modulation-demodulation，image processing and other fields. Applying Coordinate Rotation Digital Computer（CORDIC）algorithm to square root calculating is a new application. However，the accuracy of the basic CORDIC algorithm must be guaranteed by the times of iterations，and a larger number of iterations will cause problems such as excessive delay. By using methods such as establishing look-up tables，one-way rotation，merging iterations，and eliminating compensation factors，an improved CORDIC algorithm that can eliminate most of the iterative operations is proposed for square root calculations. Compared with the basic algorithm to calculate the square root，the improved algorithm uses half the clock cycle to get the output result，greatly reduces the output delay，and can achieve high calculation accuracy，which is more suitable for applications with high real-time requirements.

Key words：coordinate rotation digital calculation（CORDIC）；square-root calculating；one-way rotation；merging iterations；digital computers

---

* **收稿日期**：2021-01-31
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 Integrated Circuits（JCKY2019210C058）
作者简介：侯强（1971—），男，山西长治人，中国地质大学（武汉）副教授，博士
† 通信联系人，E-mail：1712430395@qq.com

开平方运算是一种应用范围广泛的数学运算，比如通信信号解调时计算信号包络需要进行开平方运算，正交调制信号提取相位信息时也需要开平方运算，它也是很多数字校正算法如功率放大器数字预失真参数提取算法中的关键运算[1]. 而平方根运算的精度和速度是开平方电路的主要性能指标. 坐标旋转数字计算机(Coordinate Rotation Digital Computer，CORDIC)算法[2-3]是提高平方根运算精度和速度的一种新颖方法，CORDIC算法在其计算过程中只涉及移位操作和加减操作，因此非常适合硬件特别是FPGA实现[4]. 该方法最初用于进行三角函数求值和产生正余弦波形，经过一定推广后也可用于计算双曲线函数[5]，采用CORDIC算法在双曲线旋转下的向量模式，可以计算平方根.

文献[6,7]对基本CORDIC算法计算平方根进行了详细阐述，它是一种循环迭代的计算方法，通过迭代运算不断逼近所要旋转的角度. 但由于迭代次数过多存在时延偏大的缺陷，同时每次迭代方向必须等待上一次迭代结束，由上次迭代剩余角度符号决定本次迭代旋转方向，限制了算法的运算速度. 文献[8]虽然对模校正因子进行了简化，但最终精度稍有损失，另外还有其他一些改进方法求平方根.

本文在前人对CORDIC算法进行优化基础上提出了一种改进算法求取平方根，将查找表法、单向旋转、合并迭代和基本CORDIC算法相结合，只需进行单向迭代运算，避免了每次旋转方向的不确定，消去了缩放因子[9]，从而有效提高了算法的工作速度. 同时把迭代运算划分为两个阶段完成：第一阶段迭代通过移位运算和减法运算直接实现，第二阶段迭代通过简化蝶形递归运算一步完成. 这样大大减少了迭代运算的次数，降低了延时，特别适合实时性要求高的应用场合. 本改进算法在XILINX公司xc6vlx75t-3ff484型号FPGA芯片进行了验证，结果表明：在保证与基本CORDIC算法精度相同的情况下，能够有效计算平方根，不但显著减少了算法迭代次数，还有效提高了算法运算速度，本改进算法应用在计算平方根方面的综合性能有了较明显提升和改善.

1 基本CORDIC算法

CORDIC算法最早是由J.Volder提出的，它是一种只需通过移位-相加运算不断迭代逼近目标值的计算方法[10]. 在XY坐标平面上将向量(x_0, y_0)旋转角度θ后得到向量(x_1, y_1)，如图1所示.

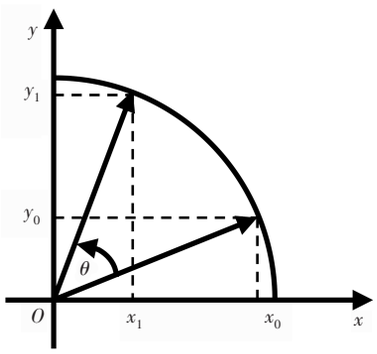


图1 向量旋转示意图

Fig.1 Vector rotation diagram

两向量间坐标变换关系如式(1)：

$$\begin{cases} x_1 = \cos\theta\cdot(x_0 - y_0\cdot\tan\theta) \\ y_1 = \cos\theta\cdot(y_0 + x_0\cdot\tan\theta) \end{cases} \tag{1}$$

如果去除$\cos\theta$项，可得到向量的伪旋转方程如式(2)：

$$\begin{cases} x_1' = (x_0 - y_0\cdot\tan\theta) \\ y_1' = (y_0 + x_0\cdot\tan\theta) \end{cases} \tag{2}$$

CORDIC算法核心在于把旋转θ角分解为N个递减的小旋转角θ_i进行N步迭代旋转，即限定旋转角度θ，使$\tan\theta=d_i\cdot 2^{-i}$，其中$d_i$=1或-1，表示旋转的方向，从而可以通过简单移位来完成由$\tan\theta$引入的乘法运算. 任意角度的旋转可通过一系列$\theta=\tan^{-1}(2^{-i})$的角度旋转迭代完成，在这里引入了角度累加器：$z_{i+1}=z_i - d_i\cdot\theta_i$用来在每次迭代过程中追踪累加后剩余的旋转角度，该剩余的旋转角度确定旋转的方向，若$z_i>0$，$d_i=1$；否则$d_i=-1$. 那么第i+1次角度的向量伪旋转方程可表示为式(3)：

$$\begin{cases} x_{i+1}' = x_i - d_i\cdot y_i\cdot 2^{-i} \\ y_{i+1}' = y_i + d_i\cdot x_i\cdot 2^{-i} \\ z_{i+1}' = z_i - d_i\cdot\theta_i \end{cases} \tag{3}$$

正如前面所述，如果消去了$\cos\theta_i$项，迭代方程(2)就只有移位和加减操作. 当$\cos\theta_i$项经过N步旋转后可得到模校正因子K_n，当N确定时K_n就是一个常量，而常数项K_n可以在系统的其他地方进行补偿，K_n表达式如式(4)：

$$K_n = \prod_{i=0}^{N-1}\cos\theta_i = \prod_{i=0}^{N-1}\frac{1}{\sqrt{1+2^{-2i}}} \tag{4}$$

最终旋转表达式如式(5)：

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = K_n\left(\prod_{i=0}^{N-1}\begin{pmatrix} 1 & -d_i2^{-i} \\ d_i2^{-i} & 1 \end{pmatrix}\right)\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{5}$$

CORDIC算法也可以用于投影计算,当将向量(x,y)投影到x轴时,此时旋转方向由y_i确定. 若$y_i<0$,$d_i=1$;否则$d_i=-1$. 迭代的最终值为式(6):

$$\begin{cases} x_n = K_n[\sqrt{x_0^2 + y_0^2}] \\ y_n = 0 \\ z_n = z_0 + \tan^{-1}(\dfrac{y_0}{x_0}) \\ K_n = \Pi\sqrt{1 + 2^{-2i}} \end{cases} \tag{6}$$

扩展迭代方程式,CORDIC算法可以用于计算双曲线方程,扩展后的向量伪旋转方程为式(7):

$$\begin{cases} x'_{i+1} = x_i - y_i \cdot \tanh(2^{-i}) \\ y'_{i+1} = y_i + x_i \cdot \tanh(2^{-i}) \\ z'_{i+1} = z_i - 2^{-i} \end{cases} \tag{7}$$

对于平方根运算采用的是双曲线方程,而且迭代模式为投影模式,迭代的最终值为式(8):

$$\begin{cases} x_n = K_n[\sqrt{x_0^2 - y_0^2}] \\ y_n = 0 \\ z_n = z_0 + \tan^{-1}(\dfrac{y_0}{x_0}) \\ K_n = \Pi\sqrt{1 + 2^{-2i}} \end{cases} \tag{8}$$

如果要求值a的平方根,只需要将x、y分别赋值为$a+1$和$a-1$,带入式(8)可得$x_n = 2\sqrt{a}$,再将其除以2即为$\sqrt{a}$.

2 改进的低时延CORDIC算法

2.1 确定旋转方向

低时延CORDIC算法核心在于确定了每次迭代的旋转方向,这样就使合并迭代成为可能. 与基本CORDIC算法令每次旋转角度为$\theta=\tanh^{-1}(2^{-i})$不同,这里令每次旋转角度为$\theta=2^{-i}$. 任意角度的旋转可通过一系列$\theta=2^{-i}$的角度旋转迭代完成,总旋转角度为$\tanh^{-1}\left(\dfrac{x_0}{y_0}\right)$,将总旋转角度使用二进制表示,若二进制数为1,则进行$\tanh(2^{-i})$的迭代旋转;否则不进行第i次迭代旋转. 这里令x_0和y_0皆为正数,那么第$i+1$次角度的向量伪旋转方程可表示为式(9):

$$\begin{cases} x'_{i+1} = x_i - y_i \cdot \tanh(2^{-i}) \\ y'_{i+1} = y_i + x_i \cdot \tanh(2^{-i}) \\ z'_{i+1} = z_i - 2^{-i} \end{cases} \tag{9}$$

2.2 建立输入值查找表

根据输入的x_0和y_0建立反双曲正切查找表[11]. 查找表中存储对应的以15位二进制数表示的$\tanh^{-1}\left(\dfrac{x_0}{y_0}\right)$值,用来作为旋转迭代的方向.

在FPGA中不能直接求取总旋转角度$\tanh^{-1}\left(\dfrac{x_0}{y_0}\right)$,所以通过MATLAB首先建立关于$\tanh^{-1}\left(\dfrac{x_0}{y_0}\right)$的查找表,由于$y_0$和$x_0$之间的关系是$x_0=y_0+2$,所以可以通过输入的$y_0$值确定总旋转角度在该查找表中的相对位置,从而在FPGA中输出$\tanh^{-1}\left(\dfrac{x_0}{y_0}\right)$.

进行角度编码确定旋转方向并不占用硬件资源,只是使每次迭代方向由输入角二进制表示时的各位的位值直接确定,避免了CORDIC基本算法中迭代方向需由剩余角度计算结果决定的不足[12],从而提高了CORDIC算法的运行速度.

基于此,可以将角度预处理后得到的二进制补码表示的15位角度值分两个阶段处理. 第一阶段,使用查找表得到的旋转方向进行单向旋转,通过移位运算和减法运算直接得到结果. 此时未旋转的角度只剩7至15位的小数部分,在第二阶段直接进行合并迭代.

2.3 单向旋转

由于FPGA中不能直接求取$\tanh(2^{-i})$,所以这里通过移位运算求取. 首先在MATLAB中求得$\tanh(2^{-i})$具体值后,如表1所示,将其转换为二进制. 此时不需要再预先存储θ_i的值,同时省去剩余角度存储,直接根据输入二进制的前6位进行处理. 可以将原来的双向旋转表达式化为单向旋转,对应输入二进制数的相应位若为1时,就取$d_i=-1$进行顺时针旋转,若为0,则不旋转直接传递x、y的值[13],即保证了精度要求又使得最终的剩余旋转角为0. 这样先根据输入角度前$(N-1)/2$位进行直接旋转,然后进行一步合并迭代运算便可求得平方根.

2.4 合并迭代

根据基本CORDIC算法有迭代公式(10):

$$\begin{cases} x'_{i+1} = x_i - y_i \cdot 2^{-i} \\ y'_{i+1} = y_i - x_i \cdot 2^{-i} \end{cases} \tag{10}$$

进行两步迭代有式(11):

$$\begin{cases} x'_{i+2} = x_{i+1} - y_{i+1}\cdot 2^{-i-1} = \\ \quad (x_i - y_i\cdot 2^{-i}) - (y_i - x_i\cdot 2^{-i})\cdot 2^{-i-1} = \\ \quad x_i - y_i\cdot 2^{-i} - y_i\cdot 2^{-i-1} + x_i\cdot 2^{-2i-1} \\ y'_{i+2} = y_{i+1} - x_{i+1}\cdot 2^{-i-1} = \\ \quad (y_i - x_i\cdot 2^{-i}) - (x_i - y_i\cdot 2^{-i})\cdot 2^{-i-1} = \\ \quad y_i - x_i\cdot 2^{-i} - x_i\cdot 2^{-i-1} + y_i\cdot 2^{-2i-1} \end{cases} \tag{11}$$

表1 旋转角度对照表

Tab.1 Rotation angle comparison table

i	$\cosh\theta_i$	$\tanh\theta_i$	15位定点小数表示$\tanh\theta_i$值
1	1.127 625	0.462 117 157	011 101 100 100 110
2	1.031 413	0.244 918 662	001 111 101 011 001
3	1.007 822	0.124 353 001	000 111 111 101 010
4	1.001 953	0.062 418 746	000 011 111 111 101
5	1.000 488	0.031 239 831	000 001 111 111 111
6	1.000 122	0.015 623 728	000 000 111 111 111
7	1.000 030	0.007 812 341	000 000 011 111 111
8	1.000 007	0.003 906 230	000 000 001 111 111
9	1.000 001	0.001 953 122	000 000 000 111 111
10	1.000 000	0.000 976 562	000 000 000 011 111
11	1.000 000	0.000 488 281	000 000 000 001 111
12	1.000 000	0.000 244 140	000 000 000 000 111
13	1.000 000	0.000 122 070	000 000 000 000 011
14	1.000 000	0.000 061 035	000 000 000 000 001
15	1.000 000	0.000 030 517	000 000 000 000 000

可见,当$i\geqslant(N-1)/2$时,基本算法中的蝶形迭代结构便可完全由一个移位-连加结构替代. 低时延算法在确定了迭代的旋转方向后,对于输出小数位宽为N的系统,根据泰勒展开式$\tanh(2^{-i})=2^{-i}-1/3\cdot 2^{-3i}+2/15\cdot 2^{-5i}-\cdots$,在$i\geqslant N/3$时可作$\theta_i=\tanh(2^{-i})\approx 2^{-i}$的近似前提下,可直接由图2所示的移位-连加的合并迭代结构替代基本算法中的蝶形迭代结构,而当$i<(N-1)/2$时可直接通过移位运算得到迭代结果. 通过观察表1中的弧度值,对于15位二进制小数表示的弧度,当$i\geqslant N/3$即$i\geqslant 5$时,$\theta_i=\tanh(2^{-i})\approx 2^{-i}$,印证上述推导结论[14].

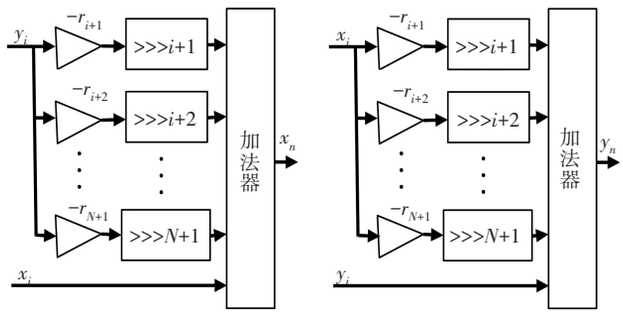


图2 合并迭代结构图

Fig.2 Merged iteration structure diagram

2.5 免除补偿因子

在CORDIC算法中$\cosh\theta_i$可由泰勒级数展开如式(12):

$$\cosh\theta_i = 1 + 2^{-2i-1} + 1/3\cdot 2^{-4i-3} + \cdots \tag{12}$$

则当$i\geqslant(N-1)/2$时,在保证系统精度的情况下$\cosh\theta_i\approx 1$. 对于15位小数系统,当$i\geqslant 7$时,迭代旋转可直接消去$\cosh\theta_i$项. 通过表1观察$\cosh\theta_i$,计算知:当$i\geqslant 7$时,$\prod_{i=7}^{15}\cosh\theta_i \approx 1.000\,040 \approx 1$,所以7到15级迭代在不进行模校正时(即免除缩放因子),在10^{-5}以上的数量级能够保证系统精度[15].

2.6 算法的FPGA实现

根据以上原理描述,用Verilog HDL语言对其进行了实现,整个程序分4个模块,分别为查找表、CORDIC算法单向旋转、CORDIC算法合并迭代、还原输出. 设计实例程序总体框图如图3.

在设计实例中,首先使用y_0进行查表,确定总旋转角度即迭代旋转次数,当$i<(N-1)/2$(即$i<7$)时,进行单向旋转. 当$i\geqslant(N-1)/2$(即$i\geqslant 7$)时,进行合并迭代.

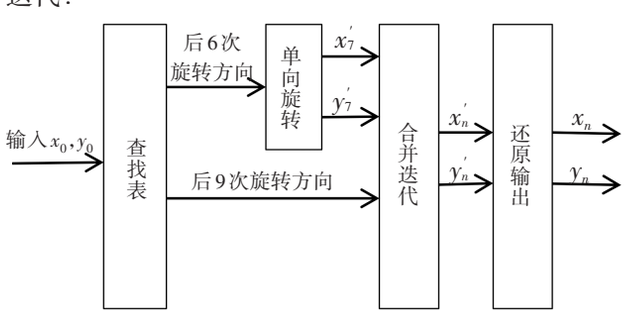


图3 设计实例总体框图

Fig.3 Overall block diagram of design example

3 仿真结果及其分析

在XILINX公司的xc6vlx75t-3ff484型号FPGA上对以上电路进行了实现,在ISE14.2软件环境下利

用其自带工具XST进行综合,并且与基本CORDIC实现方法进行了比较. 在用XST工具综合后得到电路最高工作频率和最大时延等数据,综合结果对比如表2.

表2　综合结果对比

Tab.2　Comparison of comprehensive results

CORDIC算法类型	最高运行频率/MHz	消耗寄存器数	消耗LUT数	最大输出时延
基本	127.285	705	763	16
改进	134.271	272	984	8

从表2中可以看出:改进的CORDIC算法比基本CORDIC算法的最高运行频率提高了5.49%,其原因是改进算法只需进行单向迭代运算,避免了每次旋转方向的不确定,从而有效提高了算法的工作速度. 通过对比两种算法综合后寄存器和LUT单元消耗量,可以看出改进算法相比基础算法寄存器有所减少,LUT单元使用量相对较多,但平均资源消耗两者接近. 同时改进CORDIC算法输出时延比基本CORDIC算法少了8个时钟周期,也就是节省了50%的时钟周期,其综合性能有了较大提升.

使用Xpower进行功耗分析,在40 MHz、70 MHz和100 MHz频率值上进行了测试,功率数据对比如表3. 通过表3对比得到改进CORDIC算法比基本CORDIC算法的功耗有所减少,并且随着运行频率的增加,功耗下降比率增大.

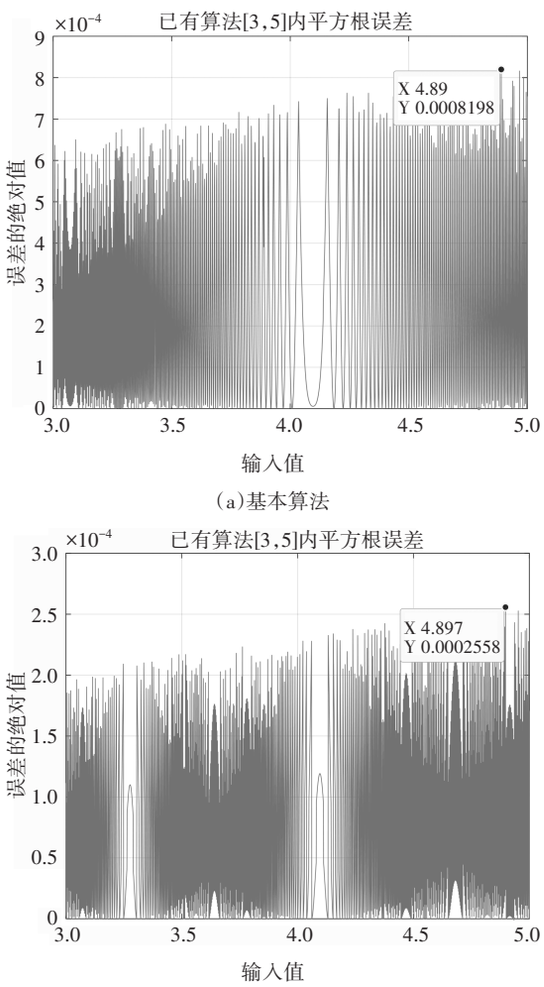
表3　功耗分析表

Tab.3　Power consumption analysis table

CORDIC算法类型 运行频率/MHz	40	70	100
基本	0.132	0.165	0.197
改进	0.131	0.162	0.191

将改进CORDIC算法和基本CORDIC算法用MATLAB语言仿真,并与理论值对比进行误差分析. 在此处为了方便观察比较,遍历求取了[3,5]部分的平方根,得到求取平方根误差的绝对值分析结果如图4.

从图4中看出:基本算法误差绝对值的最大值为8.198×10^{-4},改进算法误差绝对值的最大值为2.558×10^{-4},改进算法比基本算法在精度上提高了一些. 分别对误差绝对值进行统计平均计算,基本算法平均误差为2.435×10^{-4},改进算法平均误差为8.413×10^{-5}. 可见在平均误差上改进算法比基本算法也有所改善,主要原因在于改进算法中d_i可取0值,可以避免不必要的迭代,而基本算法对输入角度为特殊角度如θ刚好为$\tanh^{-1}2^{-i}$时仍进行迭代旋转,从而增加了算法平均误差.



(a)基本算法

(b)改进算法

图4　输出误差对比

Fig.4　Error comparison of output

4　结　论

本文针对基本CORDIC算法计算平方根中迭代次数较多,时延较长等局限提出了相应改进方法. 在保证与基本CORDIC算法精度数量级相同的情况下,减少了迭代次数,避免了每次旋转方向的不确定性,消去了缩放因子,有效降低了时延,并且最高运

行频率也有所提升.用MATLAB对该改进算法和基本算法计算平方根建模并进行了性能的比较和分析,同时在XILINX公司的xc6vlx75t-3ff484型号的FPGA上对该改进算法和基本算法计算平方根进行具体的设计和实现.仿真结果表明:改进算法输出时延减少了50%,最高运行频率提高了5.49%,并且输出精度稍优于基本算法.和基本CORDIC算法相比,改进CORDIC算法在计算平方根应用场景下的综合性能有了明显提升和改善.

参考文献

[1] FANG L L, XIE Y Z, LI B Y, *et al*. Generation scheme of chirp scaling phase functions based on floating-point CORDIC processor [J]. The Journal of Engineering, 2019, 2019(21): 7436-7439.

[2] TIWARI V, MISHRA A. Neural network-based hardware classifier using CORDIC algorithm [J]. Modern Physics Letters B, 2020, 34(15): 2050161.

[3] 姚亚峰,徐洋洋,侯强,等. 基于小容量查找表的CORDIC算法设计[J]. 湖南大学学报(自然科学版),2019,46(4):80-84.
YAO Y F, XU Y Y, HOU Q, *et al*. Implement of CORDIC algorithm with a small capacity ROM table[J]. Journal of Hunan University (Natural Sciences), 2019, 46(4): 80-84. (In Chinese)

[4] MOPURI S, ACHARYYA A. Configurable rotation matrix of hyperbolic CORDIC for any logarithm and its inverse computation [J]. Circuits, Systems, and Signal Processing, 2020, 39(5): 2551-2573.

[5] DU X H. Implementation of DDS based on CORDIC Algorithm[J]. Journal of Research in Science and Engineering, 2020, 2(8): 117-120.

[6] PILATO L, FANUCCI L, SAPONARA S. Real-time and high-accuracy arctangent computation using CORDIC and fast magnitude estimation[J]. Electronics, 2017, 6(1): 22.

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

[8] CHEN J Y, LEI Y W, PENG Y X, *et al*. Configurable floating-point FFT accelerator on FPGA based multiple-rotation CORDIC [J]. Chinese Journal of Electronics, 2016, 25(6): 1063-1070.

[9] KAVITHA M S, RANGARAJAN P. An efficient FPGA architecture for reconfigurable FFT processor incorporating an integration of an improved CORDIC and radix-2^r algorithm[J]. Circuits, Systems, and Signal Processing, 2020, 39(11): 5801-5829.

[10] 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.

[11] 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.

[12] TORRES V, VALLS J, CANET M J. Optimised CORDIC-based atan2 computation for FPGA implementations [J]. Electronics Letters, 2017, 53(19): 1296-1298.

[13] SALEHI F, FARSHIDI E, KAABI H. Novel design for a low-latency CORDIC algorithm for sine-cosine computation and its Implementation on FPGA [J]. Microprocessors and Microsystems, 2020, 77: 103197.

[14] MEHDAOUI Y, ALAMI R. DSP implementation of the Discrete Fourier Transform using the CORDIC algorithm on fixed point[J]. Advances in Modelling and Analysis B, 2018, 61(3): 123-126.

[15] MOUNIKA K, KUMAR P P, RANI K S, *et al*. Implementation of rotation and vectoring-mode reconfigurable CORDIC[J]. International Journal of Trend in Scientific Research and Development, 2018, 2(4): 1594-1602.