DOI:10.19651/j.cnki.emt.2211921

改进 CORDIC 算法实现及其在边缘检测中的应用*

吴 昊[1,2] 刘 楠[2] 丁 朋[2] 茹占强[2] 宋贺伦[2]
(1.中国科学技术大学纳米技术与纳米仿生学院 合肥 230026; 2.中国科学院苏州纳米技术与纳米仿生研究所 苏州 215123)

摘 要: 针对图像处理中的超越函数的计算,对传统的 CORDIC 算法进行研究改进,设计并实现了定点、浮点计算的硬件单元。提出两种 CORDIC 算法迭代的微旋转角度,扩展了函数计算的定义域,并采用角度编码的方式减少了三角函数计算的迭代次数。可在向量模式下实现反正切、开方以及旋转模式下正弦、余弦这 4 种超越函数的计算。定点、浮点单元均采用流水线的结构设计,可通过模式配置选择计算的函数。浮点单元采用 IEEE-754 单精度浮点数的格式,数据通路包括对阶、迭代、规格化,以 24 个时钟周期完成一次浮点数的计算。编写 SystemVerilog 平台的验证,定点计算精度最差为 10^{-3},浮点计算误差为 10^{-7},并在 FPGA 上进行板级验证,32 bit 定点数计算最大工作频率可达 243.9 MHz,相比传统的 CORDIC 算法占用的资源更小。将改进的定点 CORDIC 算法应用于图像 Sobel 边缘检测,边缘更加清晰,成像速度更快,并搭建 FPGA 图像数据采集、处理与显示系统,完成算法处理的实际验证。

关键词: 坐标旋转数字计算机算法;超越函数计算;现场可编程门阵列;边缘检测
中图分类号: TP911.73 **文献标识码:** A **国家标准学科分类代码:** 510.1

Implementation of improved CORDIC algorithm and application in edge detection

Wu Hao[1,2] Liu Nan[2] Ding Peng[2] Ru Zhanqiang[2] Song Helun[2]
(1. School of Nano-Tech and Nano-Bionics, University of Science and Technology of China, Hefei 230026, China;
2. Suzhou Institute of Nano-Tech and Nano-Bionics, Chinese Academy of Science, Suzhou 215123, China)

Abstract: The hardware units of fixed-point and floating-point calculation are designed and implemented after studying and improving the traditional CORDIC algorithm to calculate the elementary functions in image processing. Two micro rotation angles of CORDIC algorithm iteration are proposed to expand the definition domain of function calculation, and angle coding is used to reduce the number of iterations of trigonometric function calculation. Arc tangent and square root can be calculated in rotation mode, sine and cosine can be calculated in vector mode. The units of Fixed point and floating point are designed in pipeline structure, and the functions can be selected by mode configuration. The floating-point unit bases on the format of IEEE-754 single precision floating-point number. The data path contains order matching, iteration and normalization, and can be calculated once in 24 cycles. The verification of SystemVerilog platform is realized and the worst accuracy of fixed-point calculation is 10^{-3}, and floating point is 10^{-7}. The maximum working frequency of 32 bit fixed-point calculation can reach 243.9 MHz, which takes less resources than the traditional CORDIC algorithm when verificate on the FPGA. The improved fixed-point CORDIC algorithm is applied to image edge detection based on Sobel, with clearer edges and faster imaging speed. FPGA platform for image data acquisition, processing and display system is built to complete the verification of the algorithm.

Keywords: CORDIC algorithm; calculation of elementary function; FPGA; edge detection

0 引 言

在实时的信号处理以及图像处理中,需要处理大量的超越函数的计算,尤其是在图像处理中,包含大量的正余弦、平方根计算[1]。超越函数的计算精度、速度对整个系统的实时性具有很大的影响。因此,针对超越函数的实现研究具有较大的应用价值。

超越函数的实现有多种方法,主要有坐标旋转数字计

收稿日期:2022-11-02
* 基金项目:纳米真空互联试验站(2018-000052-73-01-000356)、江苏省"六大人才高峰"高层次人才项目(XYDXX-211)资助

算算法(coordinate rotation igital computer,CORDIC)[2]、多项式逼近、查表法[3]等。采用通用处理器的软件实现方式,计算周期较长[4],而硬件实现中,查表法以及多项式逼近等方式需耗费较大的资源[5],且耗费资源与精度成正比。因而多采用基于硬件实现的 CORDIC 算法。

国内外有许多学者都对 CORDIC 算法进行过相关的研究与改进。文献[6]提出了一种新的旋转缩放 MSR-CORDIC 算法,将传统算法中的旋转和缩放阶段合并,消除现有 CORDIC 算法中缩放操作的开销,但算法实现较为复杂。文献[7]提出了一种基于二进制角度重编码、免缩放因子、迭代合并技术的 CORDIC 算法,用于实现快速幅频可调的 DDS 技术,但对于传统免缩放 CORDIC 算法的迭代速度并没有改进。文献[8]提出了一种单精度浮点超越函数专用计算电路,采用 CORDIC 参数扩展算法,通过设置迭代初值方式节省硬件面积开销,该算法最大频率为 182.916 MHz,计算误差范围在 10^{-7},主要应用于图形处理器中的浮点超越函数计算,但该浮点运算至少需要 41 个时钟周期,耗时较长。文献[9]基于 Scaling-free CORDIC 算法提出了自适应编码 CORDIC 算法,降低了循环迭代次数。文献[10]提出了一种基于免缩放 CORDIC 和传统 CORDIC 算法结合的改进 CORDIC 算法。通过域折叠技术,绕过不必要的迭代步骤,减少了迭代次数和硬件资源,但并未实现统一的算法。文献[11]采用角度预处理和区间折叠,实现了高精度、低延时的 CORDIC 计算单元,但需要对角度进行预处理,增加了资源的消耗。文献[12]实现了基于改进 CORDIC 算法的 FFT 处理器的 ASIC 设计,但对于 CORDIC 算法的改进仅降低了迭代次数,并未实现完整的角度覆盖。

与此同时,针对 CORDIC 算法的应用,也有相关的一些研究。文献[13]提出了一种基于 CORDIC 的 IQ 实时解调方法,满足了雷达成像系统的需求。文献[14]提出了一种基于 CORDIC 的高速 Sobel 算法实现,实现了 Sobel 算法的加速,提高了系统的工作频率。文献[15]提出了一种机器人减速器负载的 CORDIC 计算算法,能够有效降低计算的延迟。文献[16]提出了一种基于 CORDIC 算法的血管内超声成像系统的数字坐标转换方法。但以上应用均采用传统的 CORDIC 算法,没有对 CORDIC 算法的改进展开研究。

针对以上研究中存在的问题,本文提出了两种微旋转角度实现 CORDIC 计算的角度全覆盖,并通过模式配置,可分别求解正余弦、平方根和反正切,并基于其中一种微旋转角对迭代过程中的角度进行编码,降低迭代次数为原来的一半,加快对三角函数的求解。与此用时,基于改进的 CORDIC 算法,采用流水线结构,分别设计定点、单精度浮点的硬件单元,大大降低浮点超越函数运算的时钟周期。同时,将改进 CORDIC 算法应用到基于 Sobel 算子的边缘检测中,提高边缘检测的灵敏度和速度。

1　传统 CORDIC 算法

CORDIC 算法作为一种迭代算法,仅包含移位和加法运算,被广泛用于数字信号处理,如快速傅里叶变换(FFT)、离散余弦变换(DCT)、Sobel 边缘检测等。算法可应用于圆形坐标系、直角坐标系和双曲坐标系下,包含旋转模式和向量模式。

1.1　迭代方程

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = K_i \cdot \begin{bmatrix} 1 & -\sigma_i \cdot 2^{-i} \\ \sigma_i \cdot 2^{-i} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{1}$$

该迭代方程表示迭代过程中第 $i+1$ 次的迭代结果,σ_i 代表此次迭代旋转的方向,即顺时针或者逆时针。此处 K_i 表示在伪旋转中省去的 $\cos\theta_i$。其中 $\theta_i = \tan^{-1}(\sigma_i \cdot 2^{-i})$。因此对于总的 n 次迭代来说,模长补偿因子 K 表示为:

$$K = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1+2^{-2i}}} \tag{2}$$

当迭代次数达到 16 次时,K 近似等于 0.607 252 935。同时,引入一个新的变量 z,定义为:

$$z = z_0 - \sum_{i=0}^{n-1} \sigma_i \cdot 2^{-i} \tag{3}$$

1.2　旋转模式和向量模式

如表 1 所示,当应用于圆形坐标系时,在旋转模式下,设置 x,y,z 的初值,判断变量 z 的正负,若 z 为正,则 $\sigma_i = +1$,若 z 为负,则 $\sigma_i = -1$,通过迭代可求解正弦和余弦函数;向量模式下,判断 y 的正负,若 y 为正,则 $\sigma_i = -1$,y 为负,则 $\sigma_i = +1$,通过迭代可求解平方根和反正切。

表 1　旋转模式和向量模式求解应用

模式	旋转模式	向量模式
x 初值	K	x_0
y 初值	0	y_0
z 初值	θ	0
σ_i	$\mathrm{sign}(z)$	$-\mathrm{sign}(y)$
应用	$\sin\theta/\cos\theta$	$\sqrt{x_0^2+y_0^2}/\tan^{-1}(\frac{y_0}{x_0})$

当应用于直角坐标系和双曲坐标系下,可分别求出乘加、除法以及双曲余弦、双曲正弦。

1.3　角度范围

对于传统的 CORDIC 算法,其微旋转角度表示为:

$$\theta_i = \tan^{-1}(\sigma_i \cdot 2^{-i}) \tag{4}$$

其角度范围为:

$$\pm \sum_{i=0}^{\infty} \tan^{-1}(2^{-i}) = \pm 99.882^\circ \tag{5}$$

因此,在对任意角度进行计算时,需要先对角度进行判断,通过诱导公式完成结果的计算。

2　改进 CORDIC 算法设计

针对传统 CORDIC 算法角度不能实现全覆盖,需要额

外对角度进行处理，增大了资源的占用，并且传统的CORDIC算法较多的迭代次数使得计算占据较多的时钟周期，因此提出了两种角度全覆盖的微旋转角，并基于其中一种微旋转角设计用于求解三角函数的角度编码双步迭代CORDIC结构。

2.1　改进微旋转角度

当定义微旋转角度为以下两种角度时：

$$\theta_i 1 = \tan^{-1}(2^{-i+1} + 2^{-i-1}) \tag{6}$$

$$\theta_i 2 = \begin{cases} \tan^{-1}(2^4), & i = 0 \\ \tan^{-1}(\sigma_i \cdot 2^{-i+1}), & i > 0 \end{cases} \tag{7}$$

微旋转角度与对应的模长补偿因子如表2所示。

表 2　全覆盖微旋转角度

i	$\theta_i 1$	$\theta_i 2$
0	68.199°	86.424°
1	51.340°	45.0°
2	32.005°	26.565°
3	17.354°	14.036°
4	8.881°	7.125°
5	4.467°	3.576°
6	2.236°	1.790°
7	1.119°	0.895°
8	0.560°	0.448°
9	0.280°	0.224°
10	0.140°	0.112°
11	0.070°	0.056°
12	0.035°	0.028°
13	0.017°	0.014°
14	0.009°	0.007°
Sum	186.712°	186.304°
K	0.184 782 249 3	0.037 879 397 35

根据上表可知，当满足一定的迭代次数时，两种微旋转角的角度之和均大于180°，采用这两种微旋转角可实现完整定义域[－180°，180°]内的角度覆盖，减少角度扩展的步骤，降低资源的消耗。

2.2　算法误差对比

分别计算传统的CORDIC算法以及提出的这两种微旋转角度的CORDIC算法，并与MATLAB中自带的cos、sin、sqrt、arctan函数计算的结果进行对比，对误差进行分析。

设置输入的角度测试量步幅为1°，对于传统的CORDIC算法，不对输入的角度进行变换，其输入的角度范围为[－99°，99°]，算法迭代次数为16次，绝对误差为CORDIC算法计算出来的值减去MATLAB计算出来的参考值，cos、sin函数计算结果与绝对误差分别如图1、2所示，绝对误差可达到10^{-5}数量级。当采用微旋转角度为式(6)、(7)中$\theta_i 1$、$\theta_i 2$时，输入的角度测试量步幅仍为1°，输入角度范围可达到[－180°，180°]。改进的这两种CORDIC算法的cos、sin计算结果如图3所示，其中cordic_angle1表示采用微旋转角为$\theta_i 1$的CORDIC算法，cordic_angle2表示采用微旋转角为$\theta_i 2$的CORDIC算法。从图3中可以看出，改进的这两种算法能够满足全覆盖角度的计算。图4、5所示分别为改进的这两种CORDIC算法的cos、sin计算的误差，从图中可以看出计算的绝对误差均可达到10^{-5}数量级。同时，微旋转角度为$\theta_i 2$的CORDIC算法计算误差相较更小。

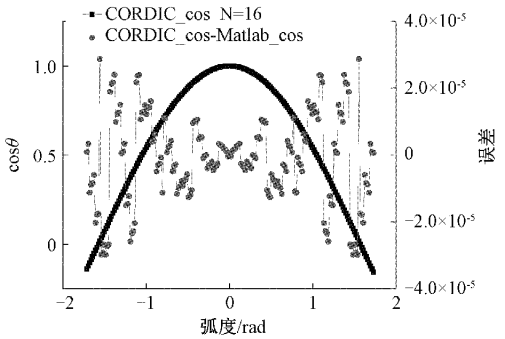


图 1　传统CORDIC算法cos计算结果与误差

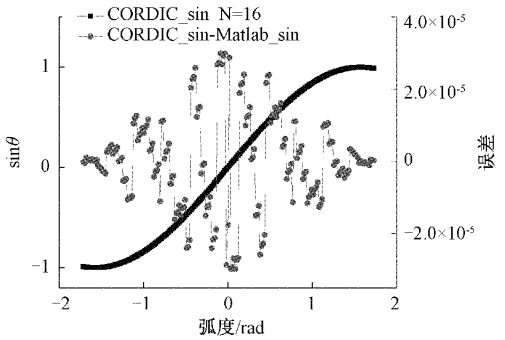


图 2　传统CORDIC算法sin计算结果与误差

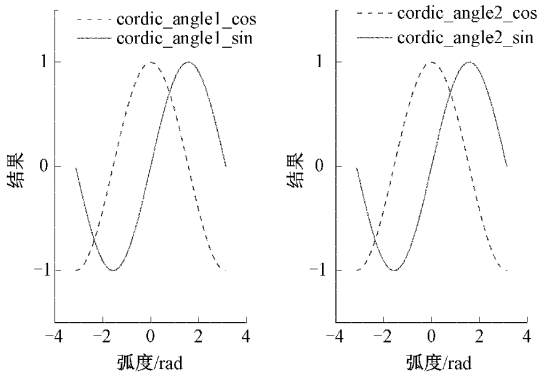


图 3　改进CORDIC算法cos、sin计算结果

当计算平方根时，测试输入为1 000以内的随机数，分别对传统的CORDIC算法、两种改进微旋转角的CORDIC算法与MATLAB自带的sqrt函数计算的差值进行对比。

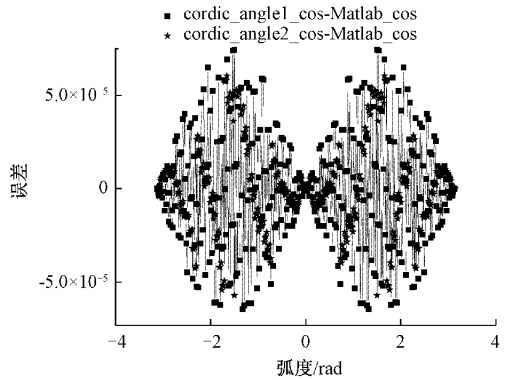


图 4　改进 CORDIC 算法 cos 计算误差

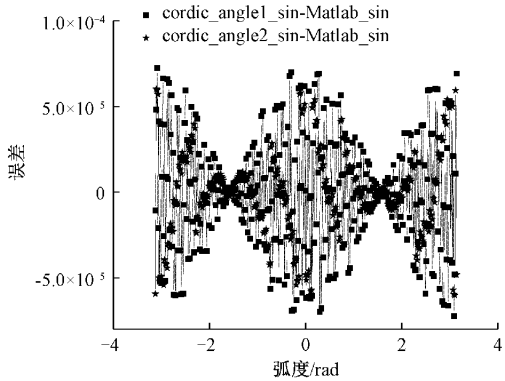


图 5　改进 CORDIC 算法 sin 计算误差

如图 6 所示,传统的 CORDIC 算法可以保持计算误差在 10^{-7} 数量级,第 1 种微旋转角度的 CORDIC 计算误差较为分散,最大不超过 4.5×10^{-6},第 2 种微旋转角度的 CORDIC 计算误差大体上落于 10^{-7} 数量级。

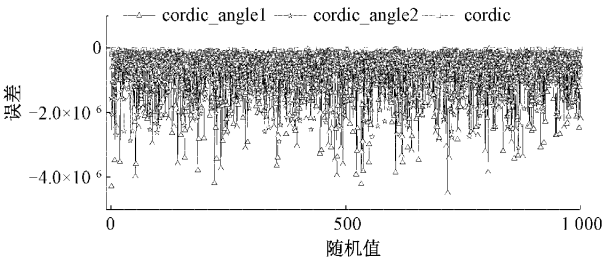


图 6　3 种 CORDIC 算法 sqrt 计算误差对比

2.3　角度编码

对于传统的 CORDIC 算法,至少需要迭代 16 次才能满足一定的精度。这是因为必须对每一次的 σ_i 进行判断后再迭代,因此增加了整体迭代的周期。

针对三角函数的求解,提出了对角度进行判断编码,从而确定后两次迭代中 σ_i 的值。如采用第二种微旋转角度 θ_i2 的 CORDIC 算法进行角度编码,实现两步迭代。表 3 为圆形坐标系下旋转模式的两步迭代方法。

采用同样的角度编码判断方法可以对 σ_i ($i=2,3,4,\cdots,15$)进行预测,使得整个迭代的次数减半,提高计算的速度。

表 3　圆形坐标系下旋转模式 σ_0、σ_1 预测

θ /(°)	σ_0	σ_1
[−180,−86.424)	−1	−1
[−86.424,0)	−1	1
[0,86.424)	1	−1
[86.424,180)	1	1

3　改进 CORDIC 算法实现

3.1　定点全覆盖 θ_i1CORDIC 结构

采用流水线的设计方式完成迭代计算的主体结构,结构示意图如图 7 所示。增加模式配置,根据 mode_cfg 的配置,可分别实现圆形坐标系下旋转模式以及向量模式的选择,在不同模式下判断 z_i 或者 y_i 的正负,决定下一步迭代的结果,分别完成正余弦、平方根和反正切的运算。

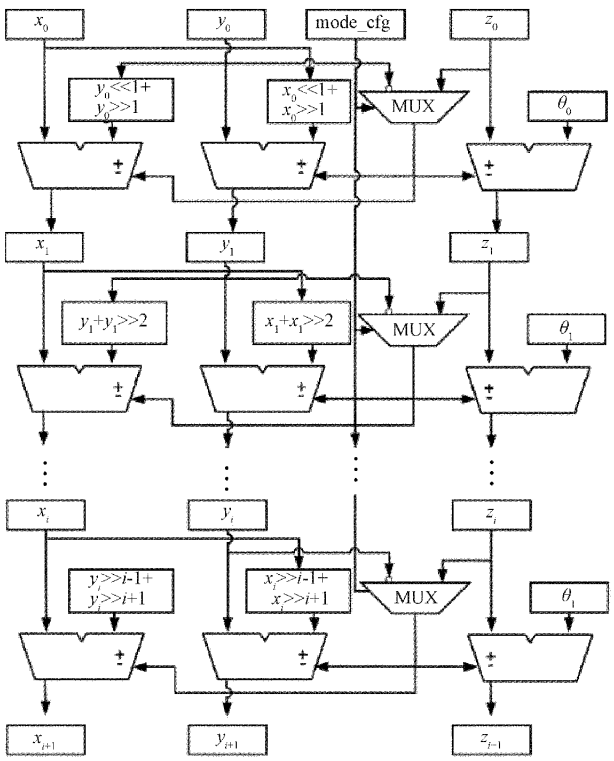


图 7　定点全覆盖 θ_i1 CORDIC 结构

Z_in[31:0] 表示输入的角度,32 bit 数的范围为 32'h0000_0000-32'hFFFF_FFFF,表示 −180°～180°,如 32'h2000_0000 表示 45°,32'hD000_0000 表示 −45°。通过最高位 Z_in[31]正负可判断角度的正负,以及最高两位数 Z_in[31:30]可判断角度所处的象限。计算过程中所有的微旋转角度事先计算好,用常量进行表示。

采用 Verilog 硬件描述语言,完成改进 CORDIC 算法的硬件实现。顶层模块接口定义如表 4 所示。

接口的时序如图 8 所示。rst_n 为复位信号,低电平有效,由低电平变为高电平后电路开始工作。X_in、Y_in、

表 4　顶层模块接口定义

接口名	位宽	in/out	功能
clk	1	in	时钟
rst_n	1	in	异步复位,低电平有效
X_in	32	in	X 路径的输入
Y_in	32	in	Y 路径的输入
Z_in	32	in	Z 路径的输入
mode_cfg	1	in	模式配置
data_ready	1	in	输入数据有效信号,高电平有效
X_out	32	out	旋转模式 sin 函数结果,向量模式 sqrt 函数结果
Y_out	32	out	旋转模式 cos 函数结果,向量模式 arctan 函数结果
data_valid	1	out	输出数据有效信号,高电平有效

Z_in 分别为 X、Y、Z 路径的输入。mode_cfg 为模式配置信号,当时钟沿到来,检测 mode_cfg 的电平进入不同的工作模式。data_ready 为输入数据有效信号,高电平有效。data_valid 表示当前模式下的函数计算结果有效,输出为高电平有效。data_ready 与 data_valid 之间的间隔为 17 个时钟周期,包括 16 次的 CORDIC 计算迭代以及最后一级输出结果的寄存。

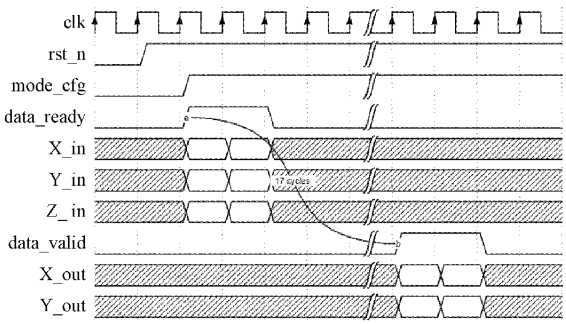


图 8　顶层模块接口时序图

3.2　定点全覆盖 θ_i2CORDIC 结构

θ_i2CORDIC 结构与 θ_i1CORDIC 结构相似,区别在于采用的微旋转角不同,使得迭代中的计算过程不同,该结构如图 9 所示,与传统 CORDIC 算法相比,增加了第一级计算,扩展了计算角度的范围,无需再对输入的角度进行处理。顶层模块的接口定义以及接口的时序与 θ_i1CORDIC 结构一致。

3.3　定点全覆盖 θ_i2 角度编码 CORDIC 结构

根据表 3 中所示的圆形坐标系下旋转模式对角度进行编码,达到两步迭代的方法,设计相应的组合逻辑电路,判断每一次迭代后的角度,计算出后两次迭代的旋转方向,使得计算三角函数时迭代次数可以缩短为原来一半。

定点全覆盖 θ_i2 角度编码 CORDIC 结构如图 10 所示。该结构的顶层接口与表 4 中所示一致,接口的时序与图 8 中相比区别为 data_ready 与 data_valid 之间的间隔为 9 个时钟周期,缩短了接近一半的时钟周期。

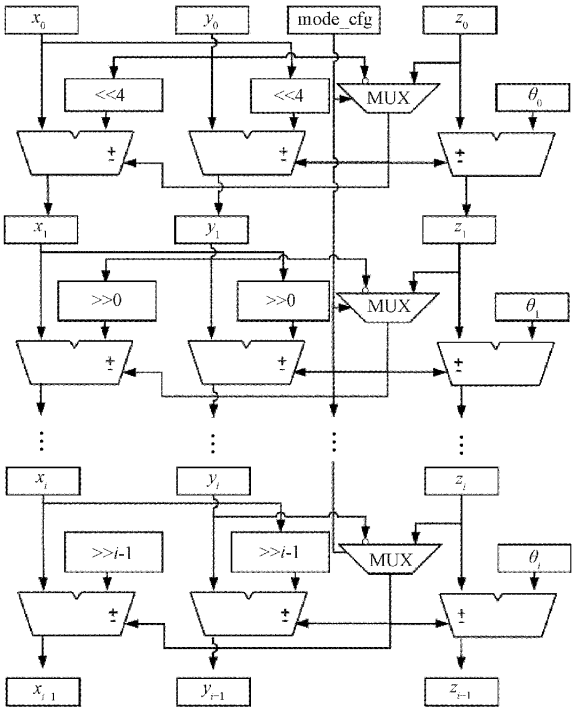


图 9　定点全覆盖 θ_i2 CORDIC 结构

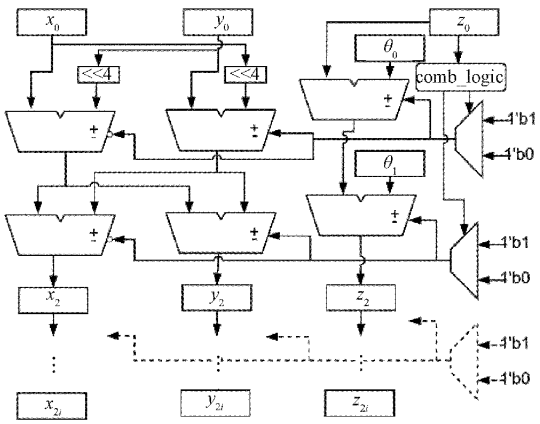


图 10　定点全覆盖 θ_i2 角度编码 CORDIC 结构

第一级组合逻辑 Verilog 代码如算法 1 所示。

算法 1:第一级组合逻辑 Verilog 代码

```
wire z0_judg0=z0 >=32'h3D74_F2BD && z0 < 32'h8000_0000;//[86.424,180)
wire z0_judg1=z0 >=32'h0 && z0 < 32'h3D74_F2BD;//[0,86.424)
wire z0_judg2=z0 >=32'hC28B_0D42 && z0 < 32'hFFFF_FFFF;//[-86.424,0)
wire z0_judg3=z0 >=32'h8000_0000 && z0 < 32'hC28B_0D42;//[-180,-86.424)
…
```

第一级时序逻辑 Verilog 代码如算法 2 所示。

算法 2:第一级时序逻辑 Verilog 代码

```
`define rot0   32'h3D74_F2BD        //86.424
`define rot1   32'h2000_0000        //45
…
case({z0_judg3,z0_judg2,z0_judg1,z0_judg0})
4'b0001: begin
   x2<=x0-(y0<<<4)-(y0+(x0<<<4));
   y2<=y0+(x0<<<4)+(x0-(y0<<<4));
   z2<=z0-`rot0-`rot1;
end
4'b0010: begin
   x2<=x0-(y0<<<4)+(y0+(x0<<<4));
   y2<=y0+(x0<<<4)-(x0-(y0<<<4));
   z2<=z0-`rot0+`rot1;
end
4'b0100: begin
   x2<=x0+(y0<<<4)-(y0-(x0<<<4));
   y2<=y0-(x0<<<4)+(x0+(y0<<<4));
   z2<=z0+`rot0-`rot1;
end
4'b1000: begin
   x2<=x0+(y0<<<4)+(y0-(x0<<<4));
   y2<=y0-(x0<<<4)-(x0+(y0<<<4));
   z2<=z0+`rot0+`rot1;
end
default: begin … end
endcase
```

通过以上逻辑的判断,能够直接根据输入的角度 z0 完成两步迭代,计算出 x2、y2、z2。那么同样地,再对 z2 的角度进行编码,判断下两步迭代的 σ_2、σ_3,能够计算出 x4、y4、z4。以此类推,直至计算出 x16、y16、z16,完成迭代。

3.4　单精度浮点全覆盖 θ_i2CORDIC 结构

针对浮点超越函数的计算,提出了一种满足 IEEE-754 格式的单精度浮点全覆盖 θ_i2CORDIC 结构,完成单精度浮点 sin、cos、sqrt、arctan 计算。

单精度浮点的数据格式如表 5 所示。

表 5　单精度浮点数据格式

32 bit	31	30:23	22:0
类型	S	E	M

注:S 表示符号位,S 为 1 表示负数,为 0 则表示正数,E 为阶数,M 表示尾数

因此,32 bit 的单精度浮点数表示的数为:

$$(-1)^S \times 1.M \times 2^{E-127} \tag{8}$$

在 CORDIC 迭代过程中包含数据的移位和加减运算,而浮点数的加减运算需要对阶数进行处理,当阶数相同才能进行尾数的计算。图 11 所示为浮点 θ_i2CORDIC 算法的数据通路,其中 S、M、E 分别表示迭代过程中的符号、尾数和阶数。由于本文采用的微旋转角能够覆盖完整的定义域,因而不需要采用参数扩展算法[8,17],只需要以下 3 个步骤。

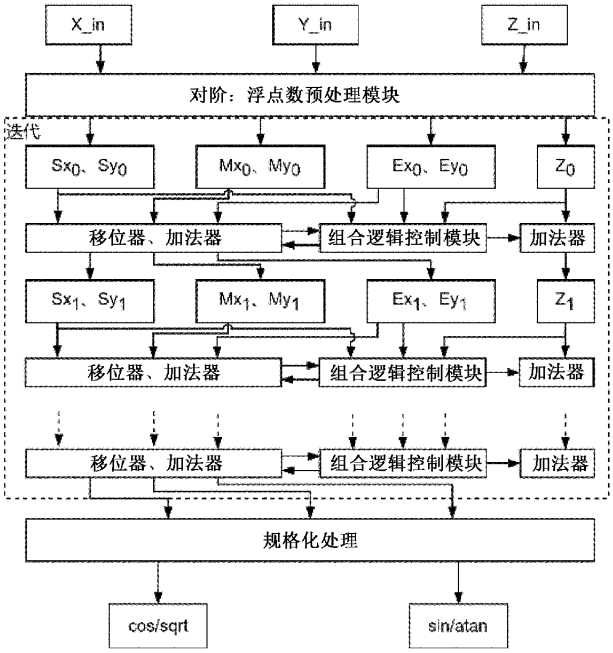


图 11　单精度浮点 θ_i2 CORDIC 数据通路

1) 对阶:提取浮点数 X_in、Y_in 的符号、阶数和尾数,比较阶数的大小。将较小阶数的浮点数的尾数进行右移,使得两浮点数的阶数一致,在计算中可实现尾数的直接计算。

2) 迭代:判断迭代过程中 Sx_i、Sy_i 和 z_i 的值,同时判断 Mx_i、My_i 移位后数据之间的大小关系,决定 Sx_{i+1}、Sy_{i-1}、Mx_{i+1}、My_{i+1}、Ex_{i+1}、Ey_{i+1} 和 z_{i+1} 的数值。

3) 规格化:对迭代后的符号、阶数、尾数进行规格化处理,将结果转换成 IEEE-754 标准的浮点数。

以上 3 个步骤中,迭代的过程较为复杂。详细过程为:首先根据当前迭代微旋转角规定的位移数,对 Mx_i、My_i 进行移位,比较 Mx_i 与 My_i 移位之后的大小关系,以及 Mx_i 移位之后与 My_i 之间的大小关系。若 x_i 和 y_i 均为正数,z_i 为负数,则 Sx_{i+1} 为 0,y_{i+1} 的正负由 Mx_i 移位之后与 My_i 的大小决定,若 Mx_i 移位之后大于 My_i,则 Sy_{i+1} 为 1,反之若 Mx_i 移位之后小于 My_i,则则 Sy_{i+1} 为 0。根据 Sx_i、Sy_i 和 z_i 为 0 或者为 1 的 8 种情况,可以确定 Sx_{i+1} 和 Sy_{i-1} 的值。

同时,根据 Sx_i、Sy_i 和 z_i 的值,判断迭代公式中做加法运算或减法运算中的两个数分别为 Mx_i、My_i、Mx_i 移位之后的值以及 My_i 移位之后的值中的哪一个。计算完之

后，若其中一数的尾数有进位，则需要阶数加一，为了保持两数的阶数相同，都使尾数向右移动一位。

4　SystemVerilog 平台验证

在软件中进行仿真，旋转模式下输入[－180°，180°]的激励，仿真波形如图 12 所示。

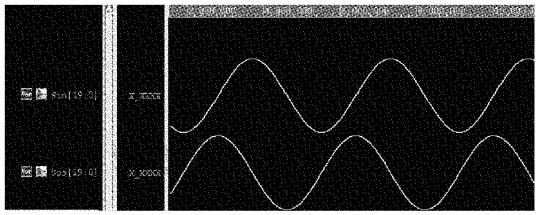


图 12　仿真波形

编写 SystemVerilog，通过“DPI-C”接口引入 C 语言的函数模型。SystemVerilog 平台发生器产生随机的数据输入到参考模型和待测试的设计，计算出来的结果分别传入计分板和监视器，而后在检查器中进行比较。SystemVerilog 验证结构如图 13 所示。

经过大规模的随机数的测试，计算过程中采用 16 bit 定点数表示小数部分，将各个算法计算出来结果与参考模型结果对比，并计算平均误差。结果通过搭建的 SystemVerilog 平台打印出来，图 14 所示为旋转模式下求解正余弦函数、参考模型的结果与误差。表 6 所示为经过大量数据测试计算的各函数的精度，即绝对误差与真值之间的比，以及算法的执行时间。

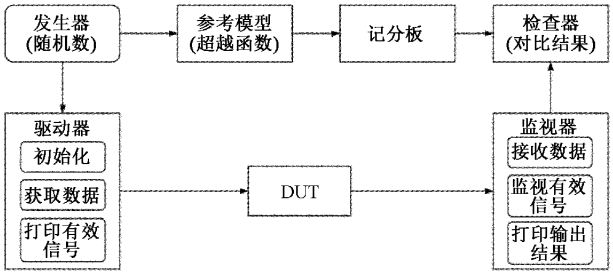


图 13　SystemVerilog 平台结构

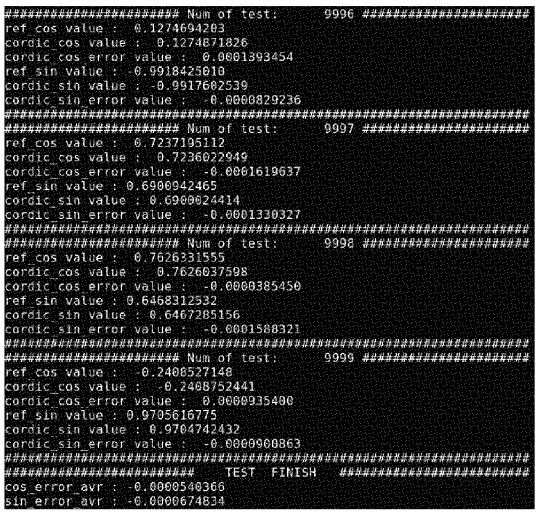


图 14　SystemVerilog 打印结果显示

表 6　各定点算法精度比较

算法	θ_i1 CORDIC	θ_i2 CORDIC	传统 CORDIC	文献[18]	θ_i2 浮点 CORDIC	文献[8]	文献[19]
数据格式	32 bit 定点	32 bit 定点	32 bit 定点	32 bit 定点	32 bit 浮点	32 bit 浮点	32 bit 浮点
正弦精度	6.7×10^{-5}	1.7×10^{-4}	2.2×10^{-4}	—	5.5×10^{-7}	1.8×10^{-7}	$<10^{-6}$
余弦精度	5.4×10^{-5}	2.5×10^{-4}	1.5×10^{-4}	—	3.1×10^{-7}	1.6×10^{-7}	$<10^{-6}$
平方根精度	1.1×10^{-3}	1.9×10^{-4}	2.9×10^{-3}	—	9.3×10^{-8}	4.6×10^{-7}	4.9×10^{-7}
反正切精度	1.9×10^{-5}	3.0×10^{-5}	1.6×10^{-5}	2.0×10^{-3}	6.8×10^{-7}	1.8×10^{-7}	—
执行时间/时钟周期	17	17	17	20	24	41－67	—

5　FPGA 综合、实现

基于 Xilinx 的 XC7K160T 型号的 FPGA 进行综合、实现。在 Vivado 软件中进行编译，各算法所占资源以及对各算法的关键路径进行计算，得出最大工作频率如表 7 所示。在 FPGA 内实现的版图如图 15 所示。

表 7　FPGA 综合、实现后性能比较

算法	θ_i1 CORDIC	θ_i2 CORDIC	θ_i2 角度编码 CORDIC	传统 CORDIC
位宽/bit	32	32	32	32
LUTs	3 666	3 647	3 320	3 762
Registers	1 671	1 681	1 020	1 678
Slice	1 016	994	917	1 022
频率/MHz	238.1	243.9	217.4	238.1

6　算法应用：基于 Sobel 算子图像边缘检测

6.1　基于 CORDIC 算法的 Sobel 边缘检测架构设计

边缘检测是图像处理和计算机视觉中的基本处理。

图 15　CORDIC 算法在 FPGA 内的布线版图

文献[14]提出了一种基于 CORDIC 的高速 Sobel 算法实现,实现了 Sobel 算法的加速,提高了系统的工作频率。本文将改进的 CORDIC 算法应用于 Sobel 算子边缘检测,结构如图 16 所示。

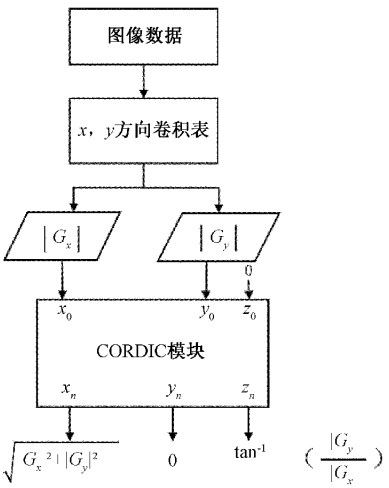


图 16　基于 CORDIC 算法的 Sobel 加速单元结构

Sobel 算子对图像 I 的水平梯度和垂直梯度的卷积分别为:

$$G_x=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\times I, G_y=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}\times I \tag{9}$$

最终图像的梯度为:

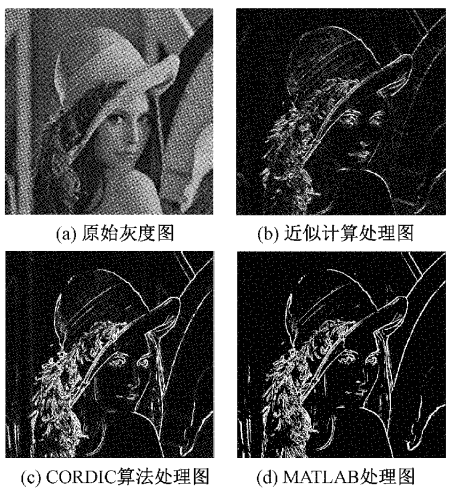
$$G=\sqrt{G_x^2+G_y^2} \tag{10}$$

梯度方向为:

$$\theta=\tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{11}$$

处理过程:读取图像数据,由于要与 3×3 的算子矩阵进行运算,需缓存 3 行的图像数据,本文中采用 3 个深度为图片宽度加 1 的 FIFO,缓存 3 行图像数据之后使能 Sobel 算子矩阵的运算,运算前在图像数据的外围进行填充操作,计算完之后送入 CORDIC 计算单元,对计算完之后的值进行阈值化处理,若梯度大于某一阈值,则认为该点为边缘点。

图 17 为边缘检测多种方法处理的结果显示图。图 17(a)为原图像的灰度图,图 17(b)为低精度近似平方根运算处理的结果图,从图中可以看出不能较好的检测出人体以及背景中的边缘信息。图 17(c)为 CORDIC 算法处理的结果图,图 17(d)MATLAB 中 Sobel 边缘检测的结果。由图 17(b)与(c)的对比可知,采用 CORDIC 算法能够更好地检测出图像的边缘,图像中的边缘细节更加丰富,并且与 MATLAB 实现的结果相似。



(a) 原始灰度图　(b) 近似计算处理图

(c) CORDIC算法处理图　(d) MATLAB处理图

图 17　边缘检测多种方法处理结果

6.2　基于 FPGA 的 Sobel 边缘检测验证

搭建 FPGA 平台的图像数据采集、处理与显示系统,架构如图 18 所示。由 FPGA 配置摄像头,获取图像数据进行图像数据的转换,并进行灰度处理,送入到基于 CORDIC 算法的 Sobel 边缘检测计算单元中。图 19 所示为搭建的 FPGA 边缘检测系统,采用 Xilinx 公司的 FPGA 型号为 XC7K160TFGG676-2,摄像头为 OV5640,显示器中分别为实际的视频显示结果以及边缘检测显示结果,从结果中可以看出边缘信息较为清晰丰富。

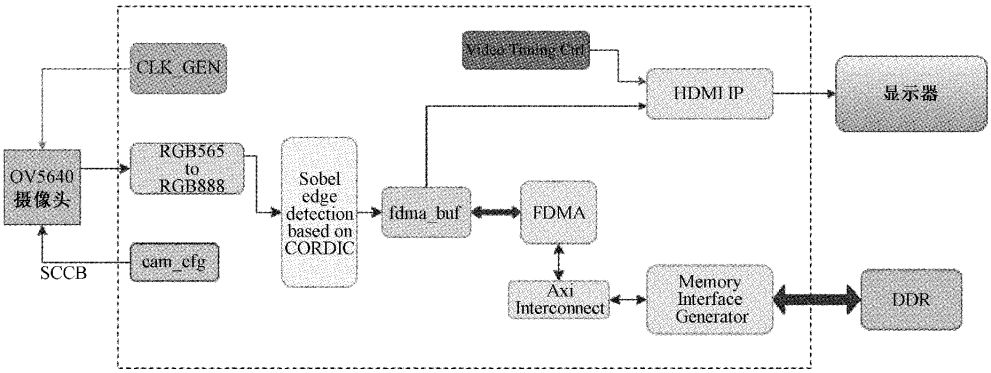


图 18　基于 FPGA 的 Sobel 边缘检测系统架构

图 19　基于 FPGA 的 Sobel 边缘检测实时显示

7　结　　论

本文针对传统 CORDIC 算法计算角度的局限性，提出了两种微旋转角度实现 CORDIC 计算的角度全覆盖，并通过模式配置，在一个模块内计算 sin、cos、sqrt、arctan，并基于其中一种微旋转角对角度进行编码，降低迭代次数，加快对三角函数的求解。完成定点以及单精度浮点的硬件单元优化设计，分别满足小面积和高精度等不同场景的需求，并在 SystemVerilog 平台和 FPGA 上分别进行了验证，提出的三种算法所占的资源均小于传统 CORDIC 算法。同时，该算法电路应用到基于 Sobel 算子的边缘检测中，能够显著提高边缘检测的灵敏度和速度。对 CORDIC 算法的研究与改进，并将该算法应用到图像处理等领域具有十分重要的意义和价值。

参考文献

[1]　杜慧敏，李萌. 面向顶点染色算法的专用指令集优化处理器[J]. 西安邮电学院学报，2014，19(2)：60-66.

[2]　VOLDER J E. The CORDIC trigonometric computing technique [J]. In IRE Transactions on Electronic Computers, 1959，8(3)：330-334.

[3]　刘小明，洪一. 基于查找表和 Taylor 展开的正余弦函数的实现[J]. 现代电子技术，2009，32(13)：165-166，170.

[4]　马士超，王贞松. 基于 DSP 的三角函数快速计算[J]. 计算机工程，2005(22)：22-24.

[5]　龙科莅. 超越函数加速器的算法设计实现[D]. 湘潭：湘潭大学，2021.

[6]　LIN C H，WU A Y. Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for high-performance vector rotational DSP applications [J]. IEEE Transactions on Circuits and Systems I：Regular Papers，2005，52(11)：2385-2396.

[7]　董宇飞，李开成，宋朝霞，等. 基于改进 CORDIC 算法的快速幅频可调 DDS 技术[J/OL]. 电测与仪表：1-9 [2023-07-25]. http://kns.cnki.net/kcms/detail/23.1202.TH.20220516.1517.004.html.

[8]　杜慧敏，沙亮，张彦芳，等. 浮点超越函数设计与实现[J]. 西安邮电学院学报，2015，(2)：16-20.

[9]　张剑锋. 基于 CORDIC 算法低功耗加速器设计与实现[D]. 长沙：国防科学技术大学，2016.

[10]　XUE Y，MA Z. Design and implementation of an efficient modified CORDIC algorithm[J]. 2019 IEEE 4th International Conference on Signal and Image Processing(ICSIP)，2019：480-484.

[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.

[12]　孔令甲. 基于一种改进的 CORDIC 算法的 FFT 芯片设计[J]. 电子技术与软件工程，2022(7)：144-147.

[13]　宋超凡，李鑫宇，简彦澎，等. 雷达成像系统中采用 CORDIC 算法的 IQ 实时解调与 FPGA 实现[J]. 电子测量技术，2020，43(18)：136-140.

[14]　黄虎，杨丁，雷宇辉，等. 基于 CORDIC 的高速 Sobel 算法实现[J]. 电子技术应用，2018，44(9)：87-90.

[15]　邓威，刘桂雄，汤少敏. 机器人减速器负载的 CORDIC 计算算法与 FPGA 实现[J]. 电子测量技术，2021，44(16)：74-78.

[16]　韩韬，邱维宝. 基于 CORDIC 算法的血管内超声成像系统数字坐标转换的 FPGA 实现[J]. 中国医疗器械杂志，2022，46(5)：485-489..

[17] 宋晨阳,李涛,牛志璐,等. 基于 CORDIC 的浮点超越函数设计与实现[J]. 信息技术,2017,(9):113-116.

[18] 高兵益,徐磊. CORDIC 算法及其展开结构的 FPGA 实现[J]. 电子测量技术,2017,40(11):85-88.

[19] LI B, FANG L, XIE Y, et al. A unified reconfigurable floating-point arithmetic architecture based on CORDIC algorithm [C]. 2017 International Conference on Field Programmable Technology (ICFPT), 2017: 301-302.

作者简介

吴昊,硕士,主要研究方向为图像处理、数字集成电路设计等。

E-mail: haow2020@mail.ustc.edu.cn

宋贺伦,博士,研究员,主要研究方向为半导体器件集成技术研究及应用。

E-mail: hlsong2008@sinano.ac.cn