DOI: 10.13382/j.jemi.B2003031

基于 HCORDIC 的浮点运算协处理器的设计*

赵 创 张 为
(天津大学 微电子学院 天津 130072)

摘 要:通信硬件、信号和图像处理上需要进行大量数学运算,坐标旋转数字计算机(CORDIC)算法可以在硬件上快速计算三角、双曲线、自然对数和平方根函数,IEEE 754 标准是目前最常用的浮点数标准,所以提出了一种处理浮点运算的协处理器。高基数自适应性 CORDIC(HCORDIC)算法具有收敛速度快的优点,通过设计用于该算法的浮点乘法器和浮点加法器,进而设计出计算多种三角函数和超越函数的浮点运算协处理器架构。该架构可以实现更快的收敛,同时减少了输出延时并具有低误差精度。设计已在现场可编程逻辑门阵列(FPGA)上实现,结果表明,相比于 Xilinx CORDIC IP 和其他 CORDIC 架构,在输出延迟、最大工作频率、关键路径和计算精度等方面有更好的表现,该设计可以应用于多种计算场景,具有较强的工程价值。

关键词: IEEE 754;FPGA;CORDIC;HCORDIC;吠陀算法;协处理器
中图分类号: TH744.3 **文献标识码**: A **国家标准学科分类代码**: 510.4030

Design of floating pointarithmetic coprocessor based on HCORDIC

Zhao Chuang Zhang Wei
(School of Microelectronics, Tianjin University, Tianjin 130072, China)

Abstract: Communications hardware, signal and image processing need a large number of mathematical operation, and coordinate rotation digital computer (CORDIC) algorithm can quickly calculate the triangle, the hyperbolic, the natural logarithm and the square root function on the hardware, what's more, IEEE 754 standard is the most commonly used floating point numbers, so proposes a processing coprocessor of floating point arithmetic. The high radix adaptive CORDIC (HCORDIC) algorithm has the advantage of fast convergence speed. By designing the floating-point multiplier and floating-point adder for this algorithm, the architecture of floating-point coprocessor is designed to calculate various trigonometric functions and transcendental functions. This architecture can achieve faster convergence while reducing output delay and keeping low error. The design has been synthesized on the field programmable logic gate array(FPGA), the results show that compared to Xilinx CORDIC IP and other CORDIC architectures, it performs better in terms of output delay, maximum operating frequency, critical path and calculation accuracy, etc. It can be widely used in many calculation scenes and has a strong engineering value.

Keywords: IEEE 754;FPGA;CORDIC;HCORDIC;vedic algorithm;coprocessor

0 引 言

IEEE 754 标准格式单精度浮点数由符号位、指数位和尾数位组成,在存储器中占用 4 bytes(32 bits)。浮点数利用浮动小数点的方法,能够表示一个范围很大的数值,具有动态范围宽的优势,因此被普遍使用。

通信硬件、信号和图像处理等众多领域需要进行大量实时数学运算,而坐标旋转数字计算机(coordinate rotation digital computer, CORDIC)能够进行大量的数学运算,如自然对数,除法,平方根,三角和双曲线变换,使用现场可编辑门阵列(field programmable gate array, FPGA)可以将计算密集型函数转换为硬件加速函数。由于 FPGA 具有并行处理的优势[1],所有密集的、迭代的计

收稿日期: 2020-03-27 Received Date: 2020-03-27
*基金项目:光电信息控制和安全技术重点实验室项目(JCKY2019210C053)、国家重点研发计划(2016YFE0100400)资助项目

算都可以在硬件上加速整个任务,例如 CORDIC 可以利用移位和加法/减法架构来获得[2],相比于软件实现,在硬件上实现简单,逻辑量小。

在电子和计算机领域中语音处理、电话信道上的数据传输、图像处理、仪器仪表、生物医学工程、地震学、石油勘探、核探测,以及处理来自外层空间的信号的应用中,离散正弦变换(discrete sine transform,DST)、离散余弦变换(discrete cosine transform,DCT)、离散傅里叶变换(discrete Fourier transform,DFT)和快速傅里叶变换(fast Fourier transform,FFT)有着不可或缺的位置,而 CORDIC 算法就是它们的基石。DCT 和 DST 使用正弦函数,正弦函数可以由 CORDIC 算法得到[3]。DCT 接近 Karhunen-Loeve 变换(Karhunen Loeve transform,KLT)的能量集中性及最佳性能使得 DCT 在语音变换和图像数据压缩的应用最为广泛。DST 广泛应用于图像重构、图像编码和图像插值。DFT 具有良好的去相关特性和能量压实特性,是通信系统、信号、语音和图像处理等各种应用中必不可少的时频转换,基于 CORDIC 的 DFT 架构在运行光谱分析、便携式数字收音机等领域得到了广泛的关注[4]。FFT 由于具有计算范围大和处理精度高的优点,在科学计算和高分辨率成像中得到了广泛的应用[5]。FFT 通过减少 DFT 所需的算术函数的数量来优化 DFT,传统 FFT 的缺点是使用了复杂的乘法器块,增加了硬件、功耗和延迟。目前已经发展了几种方法来替代乘法器,CORDIC 是目前最常用的替代 FFT 乘法器的方法。

但是传统 CORDIC 算法有迭代次数多、精度不高、收敛范围有限等缺点。为了改进 CORDIC 算法,一些研究小组引入了一些技术来减少迭代次数,包括部署更高的基数 CORDICs[6],通过查找表[7]来近似一次迭代内的目标角,以及利用多模式结构改进[8]。此外,其他方法又被提出来改善 CORDIC 性能,包括应用无比例因子的 CORDIC 来删除比例因子补偿阶段[9],使用一套新颖的角度集达到更快的收敛[10],通过预先计算微转动方向,去除每次迭代所需的符号检测[6,11]。

高基数自适应 CORDIC 算法[12]旨在探索从移位和加/减法架构到使用乘法器来增加并行性和减少迭代,利用 IEEE754 浮点格式能得到更高的精度。乘法器可以实现一个非恒定的比例因子,因此能够自适应地选择微转动。这个算法已经通过更少的微转动被证明具有更快的收敛性[12]。

1　相关理论

1.1　CORDIC 算法

CORDIC 通过将向量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 移动到 $m\in\{1,0,-1\}$ 模式的坐标系上来计算函数,其中 $m=1,0,-1$ 分别对应于圆坐标系、线性坐标系和双曲坐标系。z 表示累计角度。这个算法的特点在于将 tan 的值近似为 2^{-i},且由一个二进制移位实现,其中 i 表示迭代次数。切线定义了输入向量沿轨迹的角度增量。2^{-i} 乘以一个权重因子 $\sigma_i\in\{-1,1\}$,其值分别表示顺时针或逆时针方向旋转。CORDIC 方程如下:

$$x_{i+1}=x_i+m\sigma_i 2^{-i}y_i \tag{1}$$

$$y_{i+1}=y_i-\sigma_i 2^{-i}x_i \tag{2}$$

$$z_{i+1}=\begin{cases}z_i+\sigma_i\tan^{-1}(2^{-i}), & m=1\\ z_i+\sigma_i(2^{-i}), & m=0\\ z_i+\sigma_i\tanh^{-1}(2^{-i}), & m=-1\end{cases} \tag{3}$$

1.2　HCORDIC 算法

HCORDIC 算法用一个近似代替了 2^{-i},这个近似取决于第 i 次迭代中变量的值,再将它乘以自适应确定的权重因子。通过上述修改,算法能更快的收敛,但增加了计算权重因子 δ_i,一定程度上扩展了硬件复杂性。

使用 IEEE 754 浮点格式,符号 x_e、y_e 和 z_e 表示 x、y 和 z 的指数部分,而 x_s、y_s 和 z_s 表示其尾数。为了实现不恒定的比例因子 K_i,必须在每次迭代中更新系数 κ_i。

$$x_{i+1}=x_i+my_i\delta_i \tag{4}$$

$$y_{i+1}=y_i-x_i\delta_i \tag{5}$$

$$z_{i+1}=z_i+\theta_i \tag{6}$$

$$K_{i+1}=K_i\cdot\kappa_i \tag{7}$$

$$\kappa_i=\begin{cases}\dfrac{1}{\cos\theta_i}, & m=1\\ 1, & m=0\\ \dfrac{1}{\cosh(\theta_i)}, & m=-1\end{cases} \tag{8}$$

HCORDIC 中的两种操作模式的公式如式(9)~(12)。因为 δ_i 的 $\boldsymbol{x}$ 向量计算可以视为 δ_i 的 $\boldsymbol{y}$ 向量计算的一个子集,所以 y 的向量化操作可以实现,这样做是可以减少硬件。这些操作类似于 CORDIC 的向量和旋转操作。与 CORDIC 不同,移位的计算和角增量在这两种操作之间是不同的。

1)向量操作:

$$\delta_i=\begin{cases}\operatorname{sign}\left[\dfrac{(y_i)_s}{(x_i)_s}\right], & |y_i|\geqslant|x_i|,m\neq0\\ \left[\dfrac{(y_i)_s}{(x_i)_s}\right]\cdot2^{y_e-x_e}, & |y_i|<|x_i|,m\neq0\\ \left[\dfrac{(y_i)_s}{(x_i)_s}\right]\cdot2^{y_e-x_e}, & m=0\end{cases} \tag{9}$$

$$\theta_i=\begin{cases}\tan^{-1}\delta_i, & m=1\\ \delta_i, & m=0\\ \tanh^{-1}\delta_i, & m=-1\end{cases} \tag{10}$$

2)旋转操作:

$$\theta_i=\begin{cases}-1, & z_e>0\\ -z_s2^{z_e}, & -\dfrac{n}{2}\leqslant z_e\leqslant 0\\ -z_i, & z_e<-\dfrac{n}{2}\end{cases} \tag{11}$$

$$\delta_i=\begin{cases}\tan\theta_i, & m=1\\ \tanh\theta_i, & m=-1\end{cases} \tag{12}$$

以线性模式下的矢量操作为例,这两种算法都规定向量必须遍历一个线性路径,直到迭代到 x 轴,这个向量遍历是以角增量的形式完成的。CORDIC 中固定的角度增量使矢量在收敛之前绕 x 轴振荡,这会导致更多的迭代。HCORDIC 中的对角增量是基于给定迭代中的变量进行调整的。因此,在收敛之前,矢量不会超过 x 轴,从而可以减少迭代次数。

2 HCORDIC 算法关键单元的设计

2.1 浮点吠陀乘法器的设计

HCORDIC 中的两种操作模式中使用了乘法器和加法器,系统的性能很大程度上取决于乘法器和加法器的处理速度,尤其是乘法器,因此需要设计性能更好的结构。吠陀经典数学可以应用一个简单的算法执行乘法,避免了冗长的传统乘法结构[13]。吠陀乘法器所需要的查找表(LUTs)和 slices 的数量更小,因此降低了功耗[14]。此外,该乘法器具有重复的、规则的结构,便于设计,并且计算乘法所需的时间比其他乘法技术要少。目前的文献中有高速吠陀乘法器的 ASIC 设计[15],FPGA 上实现了一个吠陀乘法器[16],但是仅限于定点乘法。

首先利用表 1 的吠陀算法实现了一个 3×3 的吠陀乘法器,6×6 bits 乘法器是用 3×3 bits 乘法器设计的,然后使用 4 个 6×6 bits 乘法器实现 12×12 bits 乘法器。最后 24×24 bits 乘法器是需要 4 个 12×12 bits 乘法器和 2 个 24 bits 行波加法器, 24×24 bits 乘法器如图 1 所示。设计的基于 IEEE 754 的 32 bits 浮点吠陀乘法器,如图 2 所示,其中 32 bits 浮点乘法器的尾数乘法需要 24 bits 乘法器。

2.2 浮点加/减法器的设计

32 bits 浮点加/减运算按格式依旧分为 3 部分。执行加/减取决于给定输入的符号位,如果是 0,则进行加法运算,如果是 1,则进行减法运算。通过选择适当的指数,相应的尾数对齐。尾数的对齐是通过移动其中适当的尾数来实现的。然后对对齐尾数进行加/减法。最后的结果以 IEEE 754 格式输出。设计的基本架构如图 3 所示。

表 1 吠陀算法

Table 1 Vedic algorithm

```
INPUT:n bits 的乘数和被乘数
OUTPUT:2n bits 的结果
k=0
S(k):2n bits 向量初始化为 0
  for i=0 to (n-1) do
    for j=0 to i do
      S(k)=S(k)+a(i)×b(i-j)
    end for
    k=k+1
end for
for i=(n-1) to 1 do
  for j=(n-1) to i do
      S(k)=S(k)+a(i)×b(n-(i-j))
end for
k=k+1
end for
for i=0 to (k-1) do
P=P+S(i)
end for
```

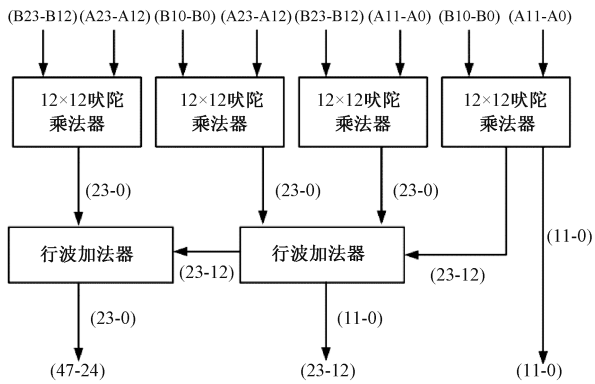

图 1 24×24 bits 吠陀乘法器

Fig. 1 24×24 bits Vedic multiplier

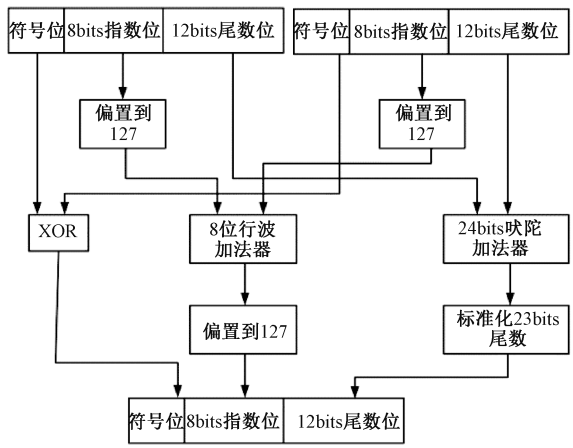

图 2 32 bits 浮点吠陀乘法器

Fig. 2 32 bits floating-point Vedic multiplier

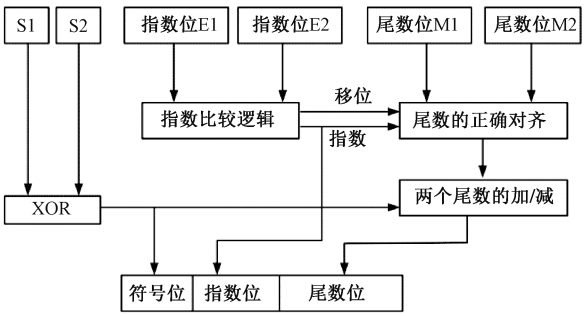

图 3　32 bits 浮点加/减法器

Fig. 3　32 bits floating-point adder/subtractor

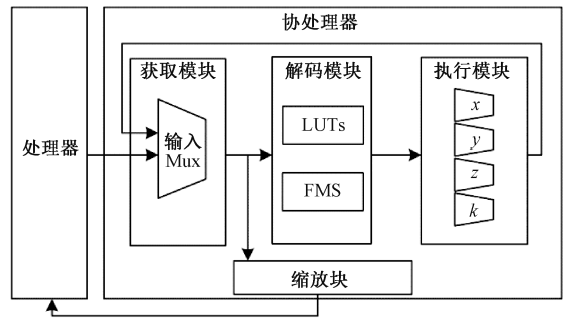

图 4　协处理器架构示意图

Fig. 4　Schematic of the coprocessor architecture

3　基于 HCORDIC 的协处理器设计

传统上 CORDIC 在 DSP 应用中的实现仅限于通用处理器上的软件实现,转移和加/减操作可以作为通用硬件数据路径的一部分实现。大多数实现的设计假设有一个固定的比例因子和一个用于固定次数迭代的状态机,ROM 是用来存储反正切值的。本文提出的硬件设计方法与上述方法有很大的不同,这是由于以下的差异:

1)非恒定比例因子,如式(7)和(8)所示,每次迭代要更新比例因子,这需要数据路径中有乘法器,乘法器的硬件成本通过迭代的快速收敛得到补偿。κ_i 的值如式(8)所示,在设计中以 ROM 的形式存储。

2)用式(9)和(11)逼近切线,2^{-i} 的更新被更复杂的计算所取代,并且没有使用移位和加法架构。由于近似值的基数不再保证是 2 的倍数,因此需要使用乘法器来执行式(4)~(7)。此外,在每次迭代中计算式(9)和(11)中给出的表达式在计算上会消耗很多资源,因此使用 LUTs 包含所有可能的输入值,为运算单元(ALU)提供 δ_i、θ_i 和 κ_i 正确的值,来执行迭代更新式(4)~(7)。

3)使用 IEEE 754 浮点数格式,IEEE 754 浮点格式的使用是 HCORDIC 算法的关键,因为它可以让运算结果实现更精确的近似。

该设计是一个具有反馈路径的 12 阶的深流水线架构。由于它作为一个协处理器,所以有对应于算法执行的各个函数的操作码。读取阶段为除平方根和自然对数之外的所有指令贡献了流水线的 2 个阶段。解码阶段占用了 2 个阶段,执行阶段因为使用浮点算术逻辑单元,所以需要 8 个阶段,这构成了一个 12 阶的流水线。从概念上看,协处理器可以分为 4 个主要阶段,即读取、解码、执行和缩放块。顶层的设计示意图如图 4 所示。

协处理器输入包含 3 个 32 bits 的变量 x、y 和 z。还提供了一个 4 bits 操作码,用于定义设计要执行的指令,以及一个 8 bits 指令标记,用于同步无序执行。输出包含 3 个 32 bits 宽的变量 x、y 和 z,以及一条指令确认信号,该指令向处理器发出信号,表明指令是否被模块接受,当流水线在反馈路径被填满时,它将降低。当模块有效输出时,输出就绪信号被设置为高,来向处理器发出信号。表 2 为不同指令对应的操作码、模式和操作。

表 2　函数的模式和操作码

Table 2　The mode and opcode of the function

操作	函数	操作码	模式
1	sin/cos	0000	圆周
(旋转操作)	sinh/cosh	0001	双曲线
0 (向量操作)	arctan	0010	圆周
	arctanh	0011	双曲线
	exp	0100	双曲线
	sqrt	0101	双曲线
	In	1001	双曲线

3.1　读取阶段

读取阶段分为预处理、预解码和仲裁 3 个阶段,如图 5 所示。

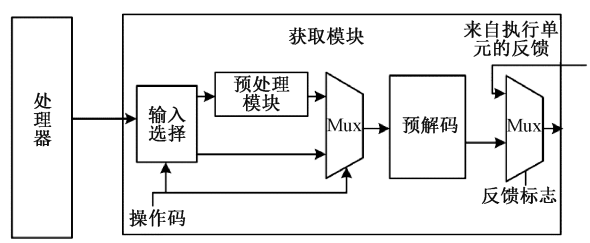

图 5　读取模块框图

Fig. 5　Block diagram of fetch module

首先是预处理,由于平方根和自然对数无法直接进行解码,因此要先对平方根和自然对数进行预处理,其他函数类型不需要预处理。根据表 3 可知,在双曲矢量操作中,可以通过设置 $x=(a+1)/2$ 和 $y=(a-1)/2$ 来计算 a 的平方根。

$$\tanh^{-1}a = 1/2[\ln(1+a) - \ln(1-a)] \qquad (13)$$

表 3 向量操作中的 CORDIC 收敛

Table 3 CORDIC converges in vector operations

模式	x	z
$m=1$	$K_1\sqrt{x_0^2+y_0^2}$	$z_0+\tan^{-1}(y_0/x_0)$
$m=0$	x_0	z_0+y_0/x_0
$m=-1$	$K_{-1}\sqrt{x_0^2-y_0^2}$	$z_0+\tanh^{-1}(y_0/x_0)$

根据表 4 可知,当 $x=(a+1)$ 和 $y=(a-1)$ 时,输出将收敛于缩放后的 $\ln a$。预处理包含一个浮点加法器,它根据给定的 a 计算 x 和 y 的值,如式(13)所示。与这些指令对应的操作码输入的额外计算将导致无序执行,附加到每一个的指令标记使处理器能够将输出映射回相应的输入。

表 4 旋转操作中的 CORDIC 收敛

Table 4 CORDIC converges in rotation operations

模式	x	y
$m=1$	$K_1(x_0\cos z_0-y_0\sin z_0)$	$K_1(y_0\cos z_0-x_0\sin z_0)$
$m=0$	x_0	$y_0+x_0z_0$
$m=-1$	$K_{-1}(x_0\cosh z_0+y_0\sinh z_0)$	$K_{-1}(y_0\cosh z_0+x_0\sinh z_0)$

下一阶段是预解码。对于给定的操作码,对应的模式和操作值被确定。例如,计算 sin 和 cos 的运算的操作码为 0000,在旋转操作时将被预先解码为旋转模式,并被分配与旋转模式相同的模式和操作位,如表 1 表示。

最后一个阶段是仲裁阶段。多路复用器(MUX)用于在反馈路径和接收到的新输入之间进行选择,如果两者同时存在,前者的优先级高于后者。此外,反馈矩阵对收敛性的检验起着非常重要的作用。在旋转操作中,通过在最后一个迭代中把 θ 设置成 $-z_i$,式(11)的最终极限确保 z_{i+1} 归 0,只有当反馈 z_i 值为 0 时,对于旋转操作的 MUX 信号才收敛。在向量操作中,MUX 检查指数 y_e-x_e 是否超出了解码块中查找表支持的范围,这个范围的限制将在解码阶段被设置。当满足这一条件时,向量操作中的 MUX 信号收敛。在收敛之后,路径转到缩放块。

3.2 解码阶段

解码模块包含的关键控制逻辑的状态机(finite state machine,FSM)以及 LUTs,它们包含权重因子 δ_i 预先被计算的值,对应角 θ_i 和比例因子 κ_i 更新的变量,如图 6 所示,其中 m 为模式,op 为操作。这些 LUTs 为每种模式和操作存储不同的值。在旋转模式,θ_i 的值是基于输入,κ_i 和 δ_i 只依赖于模式。而在向量模式,δ_i 的值仅基于输入,θ_i 和 κ_i 基于模式。LUTs 被设计成这样,可以用一个公共地址来访问 κ_i、δ_i 和 θ_i 的值。

状态机中的逻辑实现了式(9)~(12)所设置条件的约束,根据模式和操作的值,可以启用适当的 LUTs,即 LUTs 中要访问的地址取决于输入的值。从式(11)可以看出,只有当 z_e 的值小于 0 且大于 $-n/2$ 时才需要访问

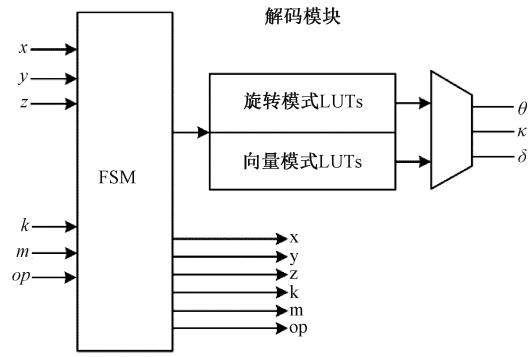

图 6 解码模块框图

Fig. 6 Block diagram of decoding module

LUTs。由式(9)可知,在非线性模式下,只有当 y_i 的值小于 x_i 时,才需要访问 LUTs。

3.3 执行阶段

执行阶段包含 4 个并行的数据路径,分别对应式(4)~(7),每个路径对应一个要更新的变量。通过观察式(4)和(5),可以看到更新 x_i 和 y_i 的两个计算是按顺序执行的,其中首先是乘法,然后是加法。式(6)和(7)表明,对 z_i 和 K_i 进行更新只需要一次计算,即对 z_i 进行加法,对 K_i 进行乘法。在本阶段中使用第 2 节设计的 IEEE 754 32 bits 浮点乘法器和加法器。

此外,在本模块引入了一个阶段来识别特殊的情况,如用于加法的 0 输入或用于乘法的 1 输入。如果检测到这种情况,则执行禁用其余的数据路径以减少功耗。

3.4 缩放阶段

当一组输入收敛时,将启用缩放块。通过与 K 相乘,x 和 y 的值被缩小到正确的输出值,此时用到了第 2 节设计的浮点乘法器。

4 结果分析

4.1 延时、最高工作频率和资源消耗分析

本文设计的协处理器的结果,已经在 Xilinx ISE14.7 中进行了模拟,并在 Xilinx Virtex 6 XC6VLX240T 上综合实现。协处理器采用 4 个模块的最长路径来计算输出延时,获取阶段使用了 3 个多路复用器逻辑、预处理模块和与预解码模块,贡献了 7 个周期延时,解码阶段的 FSM 通过输入决定必须访问的 LUT 的模式和操作,贡献了 2 个延时周期,由于执行阶段使用了浮点乘法器和浮点加法器,x 和 y 数据路径中的浮点运算单元所需的时钟周期增加了 8 个周期延时,另外还有缩放块的浮点乘法器贡献了 5 个延时周期,整个路径输出延时为 22 个周期,最大工作频率为 234.7 MHz。

以计算正余弦函数为例来比较,输入输出均为

32 bits,相比于 Xilinx CORDIC IP 字串行架构输出延时减少了 38.9%,最大工作频率提高了 17.8%,相比于 XilinxCORDIC IP 并行架构输出延时减少了 35.3%。文献[17]将常规 CORDIC 分为前后两段以减少迭代次数,并且使用移位操作代替查找表,最后得到改进的算法结构最高工作频率为 224.5 MHz,比本文设计架构的最高工作频率少了 4.5%,但该文献仅对 CORDIC 算法本身做了一些优化,没有对多种函数运算进行架构设计。文献[18]利用查找表技术和区域折叠以及双步迭代技术,提高了三角函数计算电路的性能,综合之后的输出延时为 24,比本文设计的延时多了 9.1%,与文献[17]一样,该设计仅实现了正余弦函数,未对更多种函数架构进行设计。

表 5　延时和最高工作频率对比

Table 5　Comparison of latency and operating frequency

设计方法	输出延时/周期数	最大工作频率/MHz
Xilinx CORDIC IP(字串行架构)	36	193
Xilinx CORDIC IP(并行架构)	34	–
文献[17]	–	224.5
文献[18]	24	–
本文	22	234.7

本文所设计的协处理器的资源利用率总结在表 6 列出,其中寄存器和查找表的利用率仅占 1%和 6%。

表 6　资源利用率

Table 6　Resource utilization

使用的逻辑	已用	可得	利用率/%
寄存器的数量	3165	301440	1
查找表的数量	8962	150720	6

4.2　关键路径和功耗分析

表 7 为本文设计的 4 个模块和两个对比文献的关键路径和功耗的值。分析每个模块的关键路径和功耗,通过对每个模块单独测试,然后取其平均值获得近似值,从表 7 可以看出执行单元具有最高的关键路径和最大的功耗,这是因为浮点乘法器和浮点加法器为设计提供了最高的关键路径,并且包含复杂的组合路径,使执行单元具有最高的功耗。

表 7　协处理器的模块化关键路径和功耗分析

Table 7　Modular critical path and power analysis of coprocessors

模块	关键路径/ns	功耗/mW
读取	3.443	4.218
解码	3.422	9.802
执行	4.275	38.252
缩放	3.966	1.037
本文总计	15.106	53.309
文献[19]	66.446	64.897
文献[20]	42.829	58.983

文献[19]通过对向量和旋转两种模式分别进行可重构设计,进而将两种模式进行结合实现一种广义可重构设计,提出的可重构 CORDIC 架构可用于各种应用,如同步器、波形发生器、低成本科学计算器等,文献[20]使用一种区域有效进位选择加法器(CSLA)来减少 CORDIC 实现过程中加法器数目和移位算法。从表 7 可以看到,本文设计架构的总关键路径相比文献[19-20]分别减少 339.9%和 183.5%,总功耗也有一定程度的降低。

4.3　精度与误差分析

精度和误差分析为通过选取几个角度(15°、30°、45°、60°)来测试不同输入值下本文所提出的架构的计算值和误差,其中正弦函数的计算精度和误差如表 8 所示,可以看到其计算精度达到了 10^{-8},相比于而其他文献设计的 CORDIC 和 XilinxCORDIC IP 正弦函数的计算精度为 $10^{-5}\sim10^{-6}$,计算精度有明显的提高。其中双曲正弦函数的计算精度和误差如表 9 所示,其计算精度可达到 10^{-7}。

表 8　正弦函数计算精度与误差

Table 8　Accuracy and error of sinusoidal function calculation

角度/(°)	sin 标准值	sin 计算值	误差(绝对值)
15	0.004 569 245 4	0.004 569 181 8	6.36×10^{-8}
30	0.009 138 395 4	0.009 138 468 3	7.29×10^{-8}
45	0.013 707 354 6	0.013 707 283 9	7.07×10^{-8}
60	0.018 276 027 6	0.018 275 969 3	5.83×10^{-8}

表 9　双曲正弦函数计算精度与误差

Table 9　Accuracy and error of hyperbolic sine function calculation

角度/(°)	sinh 标准值	sinh 计算值	误差(绝对值)
15	0.264 800 227	0.264 801 064	8.37×10^{-7}
30	0.547 853 474	0.547 854 333	8.59×10^{-7}
45	0.868 670 961	0.868 670 173	7.88×10^{-7}
60	1.249 367 051	1.249 365 421	1.63×10^{-6}

为了更好的验证本文设计的协处理器的精度,在[0,π/2]范围均匀取 2 000 个点,用 MATLAB 软件生成激励数据,然后分别对正余弦函数和双曲正余弦函数进行相对误差的仿真,再将数据读入生成结果图片,图 7 所示为正余弦函数运算相对误差,图 8 所示为双曲正余弦函数运算相对误差。从图 7、8 可以看到,正余弦函数运算的相对误差数量级大部分集中在 $10^{-7}\sim10^{-8}$,双曲正余弦函数运算的相对误差数量级大部分集中在 $10^{-6}\sim10^{-7}$。

根据式(14)求得平均相对误差(mean squared error,MRE),便于分析本文计算结果和标准值之间的误差。使用 MATLAB 可以得到图 7 和 8 的平均相对误差分别为 6.13×10^{-8} 和 8.26×10^{-7}。

$$\overline{E}=\sum_{n=1}^{CNT}|y\_i\_(n)-y| \tag{14}$$

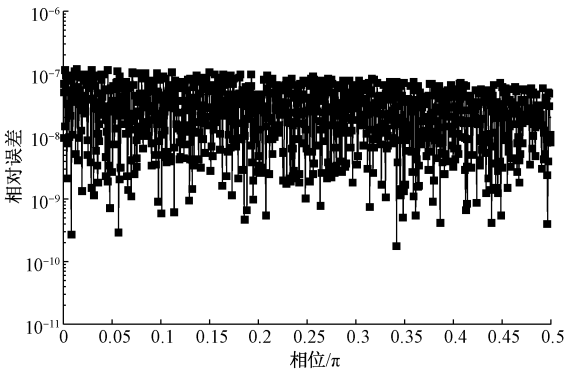

图 7 正弦函数运算相对误差

Fig. 7 Relative errors of sines and cosines

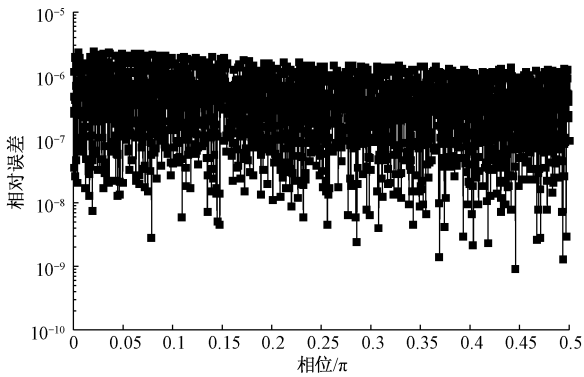

图 8 双曲正弦函数运算相对误差

Fig. 8 Relative errors of hyperbolic sines and cosines

5 结 论

本文提出了符合 IEEE754 标准的 32 bits 浮点吠陀乘法器和浮点加法器,并将其用在 HCORDIC 算法结构中,设计了基于 HCORDIC 的 12 级深流水线协处理器。通过对一个非恒定的比例因子的迭代计算,浮点乘法器被用来实现更快的收敛。解码单元被设计用于微转动的自适应选择,并且使用一个全新的查找表。4 个浮点逻辑运算单元被并行化来获得更高的吞吐量。为了适应 sqrt 和 ln 两种函数,该设计支持无序执行。使用的 IEEE 754 浮点格式提供了更高的计算精度。该设计可以实现对常用三角函数和双曲函数的计算,经过对本文设计进行仿真和综合,结果表明该设计能在输出延迟、最大工作频率、关键路径和计算精度上得到改善。

参考文献

[1] 邓澈, 颜晗, 华波,等. 基于 FPGA 的叶尖间隙信号高速采集与处理方法[J]. 电子测量与仪器学报, 2018, 32(3):104-110.
DENG CH, YAN H, HUA B, et al. High speed acquisition and processing method of blade tip clearance signal based on FPGA [J]. Journal of Electronic Measurement and Instrumentation, 2018, 32 (3): 104-110.

[2] 马峻,毛露露,陈宏,等. 数字干涉图实时相位提取 PSM-IP 核的 FPGA 实现[J]. 电子测量与仪器学报, 2019,33(12):71-79.
MA J, MAO L L, CHEN H, et al. FPGA implementation of real-time phase extraction of PSM-IP core from digital interferogram [J]. Journal of Electronic Measurement and Instrumentation, 2019, 33 (12): 71-79.

[3] JAIN A, PANDEY N, JAIN P. FPGA-based architecture for implementation of discrete sine transform [C]. Advances in System Optimization and Control, 2019: 13-22.

[4] KULSHRESHTHA T, DHAR A S. CORDIC-based high throughput sliding DFT architecture with reduced error-accumulation [J]. Circuits, Systems, and Signal Processing, 2018, 37(11): 5101-5126.

[5] MAHDAVI H, TIMARCHI S. Area-time-power efficient FFT architectures based on binary-signed-digit CORDIC[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2019, 66(10): 3874-3881.

[6] AGUIRRE-RAMOS F, MORALES-REYES A, CUMPLIDO R, et al. An Area efficient composed CORDIC architecture [J]. Advances in Electrical and Computer Engineering, 2014, 14(2): 113-116.

[7] ZHANG Y, LIAO Y T, CHEN K T, et al. Hardware architecture for CORDIC with improved rotation strategy[J]. Journal of Signal Processing, 2016, 20(4): 141-144.

[8] 刘小宁,谢宜壮,陈禾,等. 多模式 CORDIC 算法结构改进与实现[J]. 电子学报,2018,46(2):495-500.
LIU X N, XIE Y ZH, CHEN H, et al. Structure improvement and implementation of multi-mode CORDIC algorithm[J]. Acta Electronica Sinica, 2018, 46(2): 495-500.

[9] 宋定昆,刘桂雄,唐文明. 改进 SFCORDIC 算法正余弦函数求解及其应用[J]. 中国测试, 2016, 42(12): 100-104.
SONG D K, LIU G X, TANG W M. Solution and application of sines and cosines of improved SFCORDIC algorithm [J]. China Test, 2016,42(12): 100-104.

[10] GARRIDO M, KÄLLSTRÖM P, KUMM M, et al. CORDIC II: A new improved CORDIC algorithm [J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2015, 63(2): 186-190.

[11] SHABANI A, TIMARCHI S. Low-power DCT-based compressor for wireless capsule endoscopy[J]. Signal Processing: Image Communication, 2017, 59: 83-95.

[12] OZA S S. Hardware architecture for high radix adaptive CORDIC algorithm[D]. Surathkal: National Institute of Technology Karnataka, 2015.

[13] KODALI R K, BOPPANA L, YENAMACHINTALA S S. FPGA implementation of Vedic floating point multiplier[C]. IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015: 1-4.

[14] PRASADA G S S V, SESHIKALA G, NIRANJANA S. Design of high speed 32-bit floating point multiplier using Urdhva Triyagbhyam sutra of Vedic mathematics[J]. International Journal of Recent Technology and Engineering, 2019, 8(2 Special issue 3): 1064-1067.

[15] NAGARAJU M, PRAKASH R S, BHASKAR B V. High speed ASIC design of complex multiplier using Vedic mathematics[J]. International Journal of Engineering Research and Applications (IJERA), 2013, 3(1): 1079-1084.

[16] RAO K D, GANGADHAR C, KORRAI P K. FPGA implementation of complex multiplier using minimum delay Vedic real multiplier architecture[C]. IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON), 2016: 580-584.

[17] 任小西,沈建龙. 低时延-消耗的 CORDIC 算法及结构的研究[J]. 计算机科学,2014,41(8):25-29.
REN X X, SHEN J L. Study on the low delay consumption CORDIC algorithm and its structure [J]. Computer Science,2014,41(8):25-29.

[18] 李天立,尹韬,魏星,等. 高效单精度浮点三角函数计算电路结构与实现[J]. 微电子学与计算机,2018,35(12):33-37.
LI T L, YIN T, WEI X, et al. Circuit structure and implementation of efficient single-precision floating point trigonometric function calculation [J]. Microelectronics & Computer, 2018,35(12):33-37.

[19] AGGARWAL S, MEHER P K, KHARE K. Concept, design, and implementation of reconfigurable CORDIC[J]. IEEE Transactions on Very Large Scale Integration Systems, 2016, 24(4):1588-1592.

[20] INGUVA S C, SEVENTLINE J B. Enhanced CORDIC algorithm using an area efficient carry select adder[C]. 2017 International Conference on Intelligent Sustainable Systems (ICISS), IEEE, 2017: 410-415.

作者简介

赵创,2016 年于吉林大学获得学士学位,现为天津大学硕士研究生,主要研究方向为数字信号处理 VLSI 与通信系统。
E-mail:zhao1994chuang@163.com

Zhao Chuang received B. Sc from Jilin university in 2016. She is a M. Sc. candidate at Tianjin University. Her main research interests include digital signal processing VLSI and communication systems

张为,分别在 1997 年、2000 年和 2002 年于天津大学获得学士学位、硕士学位和博士学位,现为天津大学教授,博士生导师,主要研究方向为数字信号处理 VLSI 与通信系统,射频集成电路设计,图像分析与模式识别。
E-mail:tjuzhangwei@tju.edu.cn

Zhang Wei received his B. Sc., M. Sc. and Ph. D. degrees from Tianjin university in 1997, 2000 and 2002, respectively. He is now a professor and Ph. D. supervisor at Tianjin University. His main research interests include digital signal processing VLSI and communication systems, radio frequency integrated circuit design, image analysis and pattern recognition.