

面向工控 MCU 的超越函数单元设计*

宋敏特^{1,2}, 刘楠², 茹占强², 殷志珍², 丁朋², 王争光^{1,2}, 程素珍^{1,2}, 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, 江苏 苏州 215123)

(2022 年 12 月 6 日收稿; 2023 年 2 月 9 日收修改稿)

Song M T, Liu N, Ru Z Q, et al. Industrial MCU oriented transcendental function unit design [J]. Journal of University of Chinese Academy of Sciences, 2025, 42(2): 260-267. DOI: 10.7523/j.ucas.2023.009.

摘要 设计一种基于数字迭代算法的多线程、高性能、可配置的超越函数的硬件单元, 支持正弦、反正切、求模长、指数和对数的计算, 可配置 4~24 bit 定点小数精度。该设计使用 SMIC 40 nm eFlash 平台的标准单元库进行综合, 最终实现了 200 MHz 的时钟频率, 面积为 301 074 μm^2 。

关键词 超越函数; 加速器; 控制算法; 数字迭代算法; CORDIC

中图分类号: TN492 文献标志码: A DOI: 10.7523/j.ucas.2023.009

Industrial MCU oriented transcendental function unit design

SONG Minte^{1,2}, LIU Nan², RU Zhanqiang², YIN Zhizhen², DING Peng²,

WANG Zhengguang^{1,2}, CHENG Suzhen^{1,2}, SONG Helun²

(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 Sciences, Suzhou 215123, Jiangsu, China)

Abstract The calculation of transcendental functions is one of the necessary steps in industrial control algorithms. As the complexity of industrial control systems increases, calculating the transcendental function by software approximation algorithm takes up a large number of CPU cycles, compressing the computational resources of real-time control algorithms and reducing the accuracy of closed-loop control. Equipped with hardware accelerating units, industrial microcontroller unit architecture becomes the preferred solution to solve this contradiction. In this paper, a multi-threaded, high-performance, configurable transcendental function unit based on digital iterative algorithms was designed, which supports trigonometric function, exponential, and logarithmic calculations. The design was synthesized by standard cell library of SMIC 40 nm eFlash platform, resulting in a clock frequency of 200 MHz and an area of 301 074 μm^2 .

Keywords transcendental function; accelerator; control algorithms; digital iterative algorithms; CORDIC

化石能源的获取、生产和分配等方面存在诸多问题, 促使人们对可再生能源技术不断变革。

随着电力电子技术和现代控制理论的发展, 新能源发电使用的逆变器、新能源汽车使用的变频器

* 纳米真空互联试验站(2018-000052-73-01-000356)和江苏省“六大人才高峰”高层次人才项目(XYDXX-211)资助

† 通信作者, E-mail: hlsong2008@sinano.ac.cn

技术迭代加速,这些领域的闭环实时控制技术也受到更多的关注,这对微控制器(microcontroller unit, MCU)的计算性能提出更高的要求。

变频器和逆变器的 MCU 控制算法的核心要素之一在于快速完成环路控制算法参数计算,从而保证控制系统的实时性和控制精度。PID 控制的 Park/Clark 变换^[1]、NLPID 算法中的输入整形^[2]、电机旋转变压器解码角度定位^[3]等常见的控制场景都需要计算三角函数、对数、指数函数等超越函数。另外,随着技术的迭代,较为先进的控制算法和更复杂的多电平拓扑结构都需要更高频次的超越函数计算能力^[4]。但是,使用通用 CPU 执行逼近算法对上述函数进行计算会消耗大量时钟周期,延长 CPU 的响应时间。在这种情况下,开发者往往会选择降低算法复杂度或采样频率从而满足实时性要求,这种方法虽能保持系统稳定性,却牺牲了控制精度。

超越函数的计算可以采用硬件加速器实现的方案降低运算周期^[5],计算方法主要包含查表法、多项式拟合法和迭代法。单一查表法是通过映射输入数据和存储地址之间的关系,直接输出计算结果的一种方式,消耗芯片资源最少。但随着精度的提升和定义域的拓宽,表点的数量会呈现指数上升,因此只适用于低精度和小范围的计算^[6]。多项式拟合法通过构造特定形式的多项式,实现区间内的函数计算,主要包含泰勒级数展开、插值法等。多项式拟合法对于特定的函数区间具有很好的拟合精度,因此有些超越函数在较宽的输入区间会采用查表分段和拟合的方式进行组合处理^[7],进而减少表点的数量。但是这种组合方式对于实现高精度的正弦/余弦计算的乘法器、加法器和查表单元来说,芯片资源消耗仍然相当可观^[8]。迭代法是硬件实现常用的一类方法,随着迭代次数的增加,计算精度具有线性收敛特性^[9],常见算法包括牛顿拉弗森迭代算法、CORDIC (coordinate rotation digital computer)算法等。

作为数字迭代算法的一种,CORDIC 算法可以以一种简洁的方式实现包括三角函数在内的多种超越函数的计算。该算法最早由 Volder 等^[10]于 1959 年提出,主要使用场景为美国航空控制系统,是一种基于平面直角坐标系旋转的算法。1971 年 Walther^[11]提出线性系统和双曲线系统统一的 CORDIC 算法,增加了乘除、双曲函数等计算的实现。虽然现代通用 CPU 的高主频和强性能

使得 CORDIC 算法有了更多的替代方案,但至今该算法在控制、图像处理和通信领域的一些场景中仍然起到不可或缺的作用^[12]。为减少迭代的次数增强计算性能,基于角度重编码的 CORDIC 算法^[13],设置多路比较器实现的并行计算可以进一步减少迭代计算的周期数^[14]。除减少迭代以外,CORDIC 算法的硬件实现具有多种优化方式,可以使用定点数或者浮点数做迭代计算便于 CPU 进行数据读写^[15],或者应用在加速非线性算法^[16]和 FFT 算法^[17]中。

然而,在工业数字实时控制场景应用,比如电机变频器和三相逆变器的控制算法中,高通量的数据流计算的需求较少,但实时性、配置灵活性和周期可预测性需求程度较高,而单一 CORDIC 算法提供的指数和对数计算更换底数方案较为繁琐。面向工控 MCU 中的实际应用需求,本文基于超越函数的迭代算法特点,设计了一种 2 通道 8 线程的超越函数单元(transcendental function unit, TFU),支持定点小数 q1.15 和 q1.31 两种数据格式,可以实现正余弦、反正切、求模、求相角、对数和指数计算,同时具有快速响应、计算周期可预测、无中断快速读取的特性。

1 算法原理和场景需求

1.1 经典 CORDIC 算法计算三角函数

CORDIC 的原理是通过多次迭代计算完成二维平面坐标轴向量的旋转,从而逼近待求解的目标角度。相比于其他类型的算法,CORDIC 仅通过移位和加减法即可实现三角函数的运算。经典的 CORDIC 包括旋转模式和向量模式 2 种计算方式^[10]。

旋转模式将位于 X 轴正半轴上的单位向量进行逐步递减且收敛的旋转,当单位向量的旋转角等于平面直角坐标系指定角度 θ 时,其坐标值对应 $\cos\theta$ 和 $\sin\theta$ 。

在平面直角坐标系中点 (x_1, y_1) 和原点构成的向量旋转到 (x_2, y_2) ,当旋转角度为 θ 的时候,满足

$$\begin{cases} x_2 = x_1 \cos\theta - y_1 \sin\theta, \\ y_2 = x_1 \sin\theta + y_1 \cos\theta. \end{cases} \quad (1)$$

CORDIC 算法定义每一次迭代计算的角度正切值收敛,则迭代表达式可写作

$$\begin{cases} x_2 = \cos\theta(x_1 - y_1 \tan\theta) = \cos\theta(x_1 - d_1 y_1 2^{-1}), \\ y_2 = \cos\theta(y_1 + x_1 \tan\theta) = \cos\theta(y_1 + d_1 x_1 2^{-1}). \end{cases} \quad (2)$$

式中: d 表示旋转角的方向,逆时针旋转时其值为 1。对于每次迭代计算而言,不计算模长的伪旋转表达式为

$$\begin{cases} x_i = x_{i-1} - d_i y_{i-1} 2^{-i}, \\ y_i = y_{i-1} + d_i x_{i-1} 2^{-i}, \\ z_i = z_{i-1} - d_i \tan^{-1} 2^{-i}. \end{cases} \quad (3)$$

进行 m 次迭代后,可以得到

$$\begin{cases} x^{(m)} = K(x^{(0)} \cos z - y^{(0)} \sin z), \\ y^{(m)} = K(y^{(0)} \cos z + x^{(0)} \sin z), \\ z^{(m)} = 0. \end{cases} \quad (4)$$

其中: K 称为模长矫正因子或伸缩系数,随着迭代次数的增加收敛于一个固定的值。此时令 x 的初值为 $1/K$, y 的初值为 0,则迭代计算 m 次后可以同时获得 z 的正弦和余弦。

向量模式将平面直角坐标系中的某个指定向量向 X 轴旋转,最终该向量到达 X 轴。向量模式的迭代表达式和旋转模式一致,仅符号位的判定有所区别。经过 m 次迭代后,方程满足

$$\begin{cases} x^{(m)} = K(x^2 - y^2)^{1/2}, \\ y^{(m)} = 0, \\ z^{(m)} = z + \tan^{-1}(y/x). \end{cases} \quad (5)$$

CORDIC 算法的收敛域较小,由等式(1)可知

$$z \in \left[- \sum_{i=1}^m \tan^{-1} 2^{-i}, \sum_{i=1}^m \tan^{-1} 2^{-i} \right] \approx [-99.7^\circ, 99.7^\circ], \quad (6)$$

则对于平面直角坐标系中的任意角度,由于 \sin/\cos 为周期函数,因此可以将角度映射到 $[0, \pi/2]$ 区间。对于非周期的反正切函数而言,可以通过查找表的缩放实现定义域的扩展。

1.2 数字迭代法计算对数和指数

下面的方程定义了对数函数的迭代运算^[5]

$$\begin{cases} x^{(i+1)} = x^{(i)} c^{(i)} = x^{(i)} (1 + d_i 2^{-i}), \quad d_i \in \{-1, 0, 1\}, \\ y^{(i+1)} = y^{(i)} - \log_2 c^{(i)} = y^{(i)} - \log_2 (1 + d_i 2^{-i}). \end{cases} \quad (7)$$

其中 $1+d_i 2^{-i}$ 可以从查找表中读出。通过 d_i 的选择让 x 逼近 1,在第 m 次迭代运算可以得到

$$\begin{cases} x^{(m)} = x^{(0)} \prod c^{(i)} \Rightarrow \prod c^{(i)} \approx 1/x, \\ y^{(m)} = y^{(0)} - \sum \log_2 c^i = y^{(0)} - \log_2 \prod c^{(i)} \\ \approx y^{(0)} + \log_2 x. \end{cases} \quad (8)$$

通过设置 y 的初值为 0 可以计算得到 $\log_2 x$ 。从式(8)易知 x 的收敛域为

$$x \in \left[\frac{1}{\prod (1 + 2^{-i})}, \frac{1}{\prod (1 - 2^{-i})} \right] \approx [0.41, 3.45]。$$

类似地,定义迭代运算

$$\begin{cases} x^{(i+1)} = x^{(i)} - \log_2 c^{(i)} = x^{(i)} - \log_2 (1 + d_i 2^{-i}), \\ y^{(i+1)} = y^{(i)} c^{(i)} = y^{(i)} (1 + d_i 2^{-i}) \quad d_i \in \{-1, 0, 1\}. \end{cases} \quad (9)$$

选择 d_i 的值使 x 趋近于 0,经过 m 次迭代可以得到

$$\begin{cases} x^{(m)} = x^{(0)} - \sum \log_2 c^i \approx 0, \\ y^{(m)} = y^{(0)} \prod c^{(i)} = y 2^{\log_2 \prod c^{(i)}} \approx y 2^x. \end{cases} \quad (10)$$

上述迭代方程中 x 的收敛域为

$$x \in \left[\sum \log_2 (1 - 2^{-i}), \left[\sum \log_2 (1 + 2^{-i}) \right] \right] \approx [-1.24, 1.56]. \quad (11)$$

1.3 工业控制场景需求

闭环 PWM 控制是当今工业控制系统中常见的技术,主要应用于电机驱动、新能源发电并网等场景。一种典型的单环 PID 电机控制算法流程如图 1 所示。测试数据显示,采用 FPU 计算单个三角函数需要约 90 个时钟周期,在 200 MHz 主频下时间约 0.45 μ s。由于高精度工控 ADC 采样率一般为 4 M,单次采样和转换耗时 0.25 μ s,因此三角函数的计算时间长度远大于 ADC 采样和转换的时间,导致 CPU 需要较长时间等待三角函数计算结果,压缩了环内其他算法的计算时间。

工业控制场景对于超越函数计算单元的需求主要包含如下几个方面:

- 1) 更短的计算时间:总线读写时间+TFU 计算时间小于 CPU 软件计算的时间;
- 2) 可配置的精度选择:工控 ADC 的精度为 12 bit 或者 16 bit,因此 TFU 在多数应用下仅需计算半精度浮点或者 q1.15 定点小数;
- 3) 单核多线程的独立访问:高性能工控软件线程间计算相互独立,TFU 的多线程支持可以减少线程间记录状态带来的时间损耗;

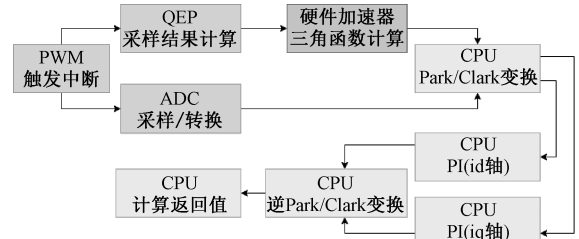


图 1 一种工控 PID 控制算法流程图

Fig. 1 A flowchart of industrial PID control algorithm

4) 双核系统的独立访问:高性能工控通常采用双核架构便于分配并行任务,因此 TFU 需要满足并行任务的计算。

基于上述分析,选用数字迭代算法实现超越函数的运算。

2 TFU 设计

2.1 TFU 算子的实现

算子即算法中的最小运算单元,定义实现每一次迭代计算的模块为一个算子。TFU 以全组合逻辑的形式实现 CORDIC 算子和指数对数 (LOGEXP)算子。

如图 2(a)所示,一个 CORDIC 算子可以实现 4 阶 CORDIC 迭代计算,每阶迭代需要从外部输入一个逐步递减的角度数据,并根据上一阶的计算结果对本阶计算的正负性进行选择。指数对数算子的电路结构如图 2(b)所示。和 CORDIC 算子类似,每个算子可以实现 4 阶指数或者对数的运算。

如表 1 所示,通过配置不同的输入参数,相同的 CORDIC 算子可以实现 5 种函数的计算,LOGEXP 算子实现 2 为底的对数和指数运算。其中反正切函数 ATAN 和求相角函数 ATAN2 实质上是同一种运算,通过配置第 2 参数 Y 为 1 可以实现 ATAN 运算,但此时第 1 输出没有实际意义。

2.2 TFU 加速器架构

工控领域常用的 ARM-M 系列处理器具有较强的多线程运算性能,为减少线程之间共用外设

和总线资源产生的运算能力下降,TFU 支持双通道计算,每个通道最高支持 4 线程数据接收,所有通道共享 AHB slave 接口。最终实现的 TFU 结构框图如图 3 所示。

其中 CSR 表示配置寄存器 (control/status register),主要实现函数不同参数的灵活配置。WDATA 和 RDATA 分别对应函数的输入和输出,均由深度为 2 级的 FIFO 实现。每个通道中,4 个线程的数据通过 AHB 总线写入后,由 ARB 模块进行仲裁,决定运算优先级,仲裁的方式可以配置为 round-robin 或者固定优先级顺序。被仲裁命中的数据将会传递到 FSM 模块中进行数据迭代,由于每次迭代的运算电路相同,因此多次迭代可以公用一组 CORDIC 或者 LOGEXP 算子,分别计算三角函数和指数、对数。此外,接口寄存器还包含 LOGEXP_LUT 域,用于 LOGEXP 算子更换底数。

FSM 控制的数据流如图 4 所示。当 AHB slave 接口接收数据后,无需额外配置使能寄存器,如果识别到数据已完成传输,状态机会直接跳转到工作模式。数据首先会转移到数据过滤器中,将多余或者出现格式错误的数据清除,随后经过数据映射模块将数据映射到算法的收敛区间。每次迭代计算的数据结果会被保存到专用的寄存器中,最后一次迭代产生的数据会输入到响应寄存器中等待总线读取。

2.3 接口与时序

TFU 顶层接口采用 AMBA3.0 AHB slave 接口,通过时序逻辑判断数据地址,将接口时序转换为寄存器读写时序连接到 4 个线程的寄存器上。但是,CPU 需要通过总线轮询或者进入中断获取各个外设状态,这 2 种类型的状态获取均需要消耗大量时钟周期,延长 CPU 获取运算结果的时间。

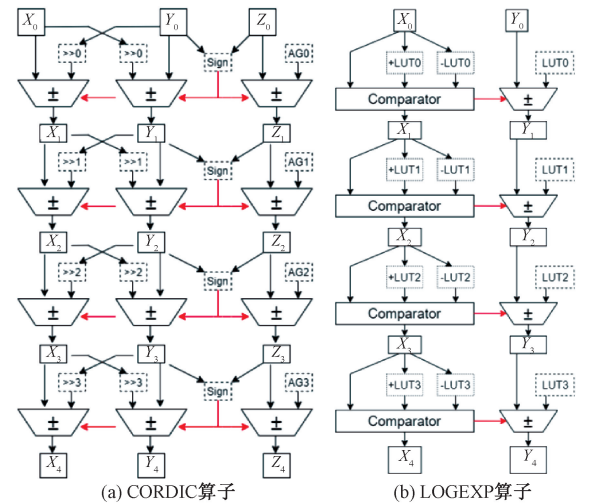


图 2 算子原理图

Fig. 2 Schematic diagram of operators

表 1 TFU 算子输入输出关系

Table 1 Input/output relationship of TFU operators

函数	SIN/COS	ATAN2/MOD	ATAN	LOG/EXP
X 输入	K	X	X	X
Y 输入	0	Y	1	1/0
Z 输入	θ	0	0	-
X 输出	$\sin\theta$	$K \times \sqrt{Y^2 + X^2}$	$K \times \sqrt{1 + X^2}$	$\text{LOG}_2(X) / 2\text{EXP}(-X)$
Y 输出	$\cos\theta$	0	0	0/1
Z 输出	0	$\text{ATAN2}(Y, X)$	$\text{ATAN}(X)$	-

数位。对于 LOGEXP_LUT 域,根据算法原理,TFU 需要 48 个查找表值,分别支持数字迭代计算中的加或者减运算,其格式满足 q1.31。在默认状态下,该底数为 2。

2.5 函数和数据映射

TFU 支持 7 种超越函数的运算,每种函数会读取最多 2 组输入数据(argument, ARG)寄存器的数据,计算结果导入到 2 组输出数据(response, RES)寄存器中。表 4 列出了各个函数的输入和输出寄存器配置要求。

表 3 数据寄存器列表
Table 3 data register table

Bit	域	类型	描述
31:0	wdata_arg0	读写	主输入数据
31:0	wdata_arg1	读写	辅输入数据
31:0	rdata_res0	只读	主输出数据
31:0	rdata_res1	只读	辅输出数据
31:0	le_lut_pls_x	读写	LOGEXP 正查找表 x
31:0	le_lut_mns_x	读写	LOGEXP 负查找表 x

表 4 TFU 函数数据寄存器描述
Table 4 Description of TFU register

函数	寄存器	描述	范围
COS	ARG0	Angle θ , divided by π	$[-1,1)$
	ARG1	Modulus m	$[0,1)$
	RES0	$m\cos\theta$	$[-1,1)$
	RES1	$m\sin\theta$	$[-1,1)$
ATAN	ARG0	$x \times 2^{-n}$	$[-1,1)$
	ARG1	-	-
	RES0	$2^{(-n)} \times \text{atan } x$, in radians, divided by π	$[-1,1)$
	RES1	-	-
PHS phase	Scale	n	$[0,7]$
	ARG0	x coordinate	$[-1,1)$
	ARG1	y coordinate	$[-1,1)$
	RES0	Phase angle θ in radians, divided by π	$[-1,1)$
Mod modulus	RES1	Modulus m	$[0,1)$
	ARG0	x coordinate	$[-1,1)$
	ARG1	y coordinate	$[-1,1)$
	RES0	Modulus m	$[0,1)$
Log	RES1	Phase angle θ in radians, divided by π	$[-1,1)$
	ARG0	$x-1$, fractional part	$[0,1)$
	ARG1	-	-
	RES0	$\text{Log}_2(x)$, base >2 , depend on LUT	$[0,1)$
Exp	RES1	-	-
	ARG0	X	$[0,1)$
	ARG1	-	-
	RES0	$2\text{EXP}(-x)$, base >2 , depend on LUT	$[0,1)$
	RES1	-	-

由于 CORDIC 和 LOGEXP 算子具有收敛域,对于域外数据或者不便处理的域内数据,函数计算需要通过数据映射的策略进行预处理。其中,ATAN 函数需要对 CSR 寄存器中的 Scale 域进行配置,从而实现对定义域外的数据进行计算。表 5 统计了 TFU 各个函数的算法收敛域、TFU 映射域和 TFU 最大可以支持的定义域。其中收敛域是指迭代计算算法原理可以支持的数据范围,三角函数用角度表示;映射域是 TFU 算子计算时的数据范围;定义域是 TFU 数据寄存器支持读取的数据范围。需要注意的是,LOGEXP 算子的定义域是对收敛域取子集,因此不需要映射域进行域扩展。

表 5 TFU 函数数据映射
Table 5 data mapping of TFU functions

函数	收敛域	映射域	定义域
SIN/COS/ATAN	$[-99.7^\circ, 99.7^\circ]$	$[0^\circ, 90^\circ)$	$[-180^\circ, 180^\circ)$
ATAN2/MOD	$[-99.7^\circ, 99.7^\circ]$	$[0^\circ, 90^\circ)$	$[-135^\circ, 135^\circ)$
LOG2	$[0.41, 3.45]$	-	$[1, 2)$
2EXP	$[-1.24, 2.56]$	-	$[0, 1)$

3 结果与数据统计

3.1 TFU 计算精度

TFU 可以通过配置 CSR 寄存器实现 1~6 时钟周期的计算,每个时钟周期完成 4 次迭代计算。图 6 展示 TFU 在不同迭代次数下最大误差的统计图。随着迭代次数的增加,函数计算的精度随之提升,并具有较好的线性度,与算法原理一致。值得注意的是,由于求模运算的结果需要乘以精度为 31 bit 的系数 K ,因此迭代次数小于 16 次时精度显著高于其他 6 个函数。各个函数的最大误差如表 6 所示。图 7 展示在工控系统中常用的 q1.15 数据格式下,迭代周期为 4 时的计算误差统计。

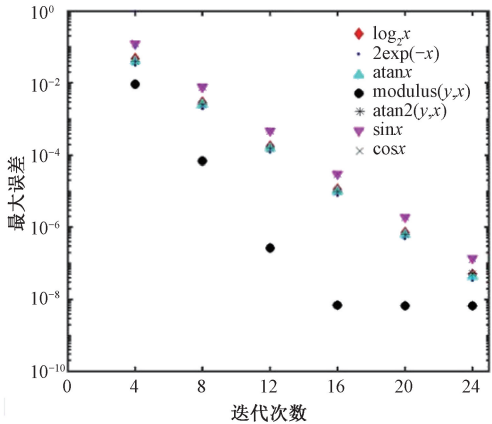


图 6 TFU 不同迭代次数最大误差统计
Fig. 6 Maximum error of TFU at different iterations

表 6 TFU 函数精度统计表

Table 6 Precision of TFU functions

迭代次数	运算周期	最大误差 q1. 15	最大误差 q1. 31
4	1	2^{-3}	2^{-3}
8	2	2^{-7}	2^{-7}
12	3	2^{-11}	2^{-11}
16	4	2^{-15}	2^{-15}
20	5	2^{-16}	2^{-19}
24	6	2^{-16}	2^{-23}

3.2 TFU 综合

TFU 通过 Synopsys Design Compiler 工具进行综合。由于 TFU 只有一个时钟域,所以设定所有的输入和输出端口的最大延迟为半个时钟周期。此外,设定 setup 和 hold 的 uncertainty 参数为 0.1 ns, transaction ratio 为 0.8。此次综合选取 SMIC 40 nm eFlash 平台的 40 nm 沟道超高速常规阈值电压(40 nm-channel very-high-speed regular threshold voltage, VHSC40-RVT)标准单元库,实现了 200 MHz 的时钟频率,实际占用面积 301 074 μm^2 。以 NAND2 标准单元估算逻辑门数量约 110 688。

为便于评估模块的规模,对 TFU 使用 Xilinx Vivado 工具进行 FPGA 综合。结果显示,在使用 Virtex7 系列 FPGA 时,TFU 共占用 LUT 数量为 18 021,寄存器数量为 6 244。

3.3 性能比较

TFU 是一种面向工控 MCU 的专用外设,具有

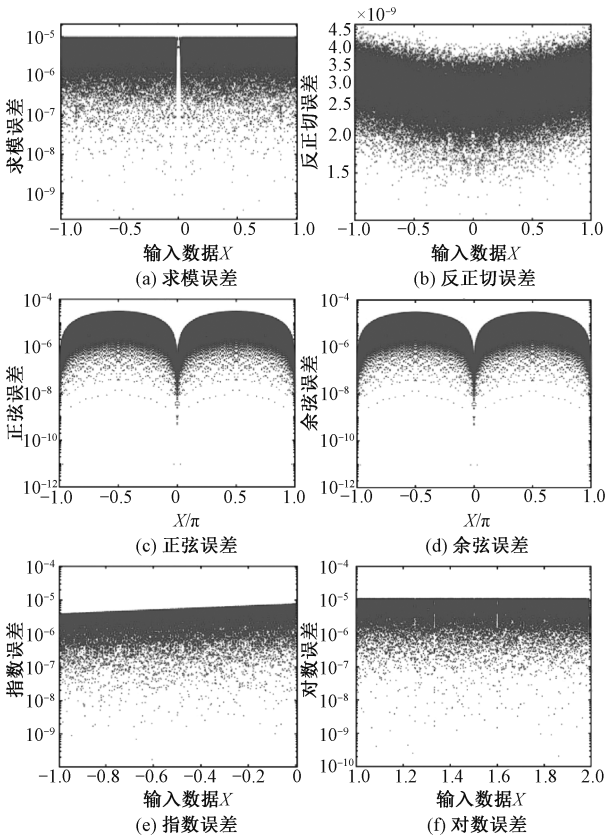


图 7 q1.15 数据计算误差统计(均为 16 次迭代)

Fig. 7 Error of TFU at q1.15 argument data format (16-times iterations)

三角函数和指数对数的运算能力。和国内外相似文献相比,TFU 具有高并行度、短计算周期、灵活配置的特性,并且提供了可以更换底数的指数对数算子。对比内容如表 7 所示。

表 7 相似功能实现的参数对比表

Table 7 Comparison of parameters of similar implementations

	周期	精度/bit	节点/nm	并行线程	频率/MHz	面积/ μm^2	LUT/FF	数据格式	支持函数
本文	3~9	4~24	40	8	200	301 074	18 021/6 244	定点小数	SIN/COS/ATAN/ PHS/LOG/EXP
文献[18]	10	10	—	1	342	—	8 428/11 168	单精度浮点	SIN/COS
文献[19]	24	23	—	1	282	—	2 810/1 993	定点小数	SIN/COS
文献[20]	3/8/11	23	65	1	55	58 000	—	单精度浮点	SIN/COS/ATAN/PHS/

4 结论

本文面向工控 MCU 的算法实时性要求,设计了一种超越函数计算外设 TFU。借助 CORDIC 算法和数字迭代计算,TFU 可以实现正余弦、反正切、求模长以及对数和指数计算,具有 4~24 bit 可配置的计算精度。此外,TFU 支持 q1.15 和

q1.31 定点小数格式,具有计算精度可配置的特性,借助总线接口可实现无中断触发的结果读取。面向双核系统架构,TFU 具有 2 通道 8 线程,和可配置的线程优先级,对数指数计算可换底数,满足多场景需求。TFU 使用 SMIC40 eFlash 标准单元库综合可以达到 200 MHz 的工作频率,综合面积为 301 074 μm^2 。

参考文献

- [1] Kang Y G, Diaz Reigosa D. Dq-transformed error and current sensing error effects on self-sensing control[J]. IEEE Journal of Emerging and Selected Topics in Power Electronics, 2022, 10 (2): 1935-1945. DOI: 10.1109/JESTPE.2021.3051942.
- [2] Texas Instruments. Digital control library[DB/OL]. (2022-04-27) [2022-12-20]. <https://training.ti.com/node/1149587>.
- [3] Ramezannezhad A, Naderi P, Vandevelde L. A novel method for accuracy improvement of variable reluctance linear resolvers[J]. IEEE Sensors Journal, 2022, 22(19): 18409-18417. DOI: 10.1109/JSEN.2022.3199807.
- [4] Rodriguez J, Lai J S, Peng F Z. Multilevel inverters: a survey of topologies, controls, and applications[J]. IEEE Transactions on Industrial Electronics, 2002, 49(4): 724-738. DOI: 10.1109/TIE.2002.801052.
- [5] Kantabutra V. On hardware for computing exponential and trigonometric functions [J]. IEEE Transactions on Computers, 1996, 45 (3): 328-339. DOI: 10.1109/12.485571.
- [6] Wang D, Muller J M, Brisebarre N, et al. (M, p, k)-friendly points: a table-based method to evaluate trigonometric function[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2014, 61(9): 711-715. DOI: 10.1109/TCSII.2014.2331094.
- [7] Tang P T P. Table-lookup algorithms for elementary functions and their error analysis[C]// [1991] Proceedings 10th IEEE Symposium on Computer Arithmetic. June 26-28, 1991, Grenoble, France. IEEE, 2002: 232-236. DOI: 10.1109/ARITH.1991.145565.
- [8] Zhu B Z, Lei Y W, Peng Y X, et al. Low latency and low error floating-point sine/cosine function based TCORDIC algorithm[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2017, 64(4): 892-905. DOI: 10.1109/TCSI.2016.2631588.
- [9] Koren I, Zinaty O. Evaluating elementary functions in a numerical coprocessor based on rational approximations[J]. IEEE Transactions on Computers, 1990, 39(8): 1030-1037. DOI: 10.1109/12.57042.
- [10] Volder J E. The CORDIC trigonometric computing technique [J]. IRE Transactions on Electronic Computers, 1959, EC-8 (3): 330-334. DOI: 10.1109/tec.1959.5222693.
- [11] Walther J S. A unified algorithm for elementary functions[C]//Proceedings of the May 18-20, 1971, spring joint computer conference. May 18 - 20, 1971, Atlantic City, New Jersey. New York: ACM, 1971: 379-385. DOI: 10.1145/1478786.1478840.
- [12] Meher P K, Valls J, Juang T B, et al. 50 years of CORDIC: algorithms, architectures, and applications [J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2009, 56 (9): 1893-1907. DOI: 10.1109/tcsi.2009.2025803.
- [13] Juang T B. Low latency angle recoding methods for the higher bit-width parallel CORDIC rotator implementations[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2008, 55 (11): 1139-1143. DOI: 10.1109/TCSII.2008.2002566.
- [14] Rodrigues T K, Swartzlander E E. Adaptive CORDIC: using parallel angle recoding to accelerate rotations [J]. IEEE Transactions on Computers, 2010, 59(4): 522-531. DOI: 10.1109/TC.2009.190.
- [15] Chandrakanth Y, Praveen Kumar M. Low latency & high precision CORDIC architecture using improved parallel angle recoding [C] // 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies. July 21-22, 2011, Thuckalay, India. IEEE, 2011: 498-501. DOI: 10.1109/ICSCCN.2011.6024602.
- [16] Mohamed S M, Sayed W S, Radwan A G, et al. FPGA implementation of reconfigurable CORDIC algorithm and a memristive chaotic system with transcendental nonlinearities [J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(7): 2885-2892. DOI: 10.1109/TCSI.2022.3165469.
- [17] Hoang T T, Nguyen X T, Le D H, et al. Low-power floating-point adaptive-CORDIC-based FFT twiddle factor on 65-nm silicon-on-thin-BOX (SOTB) with back-gate bias[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2019, 66 (10): 1723-1727. DOI: 10.1109/TCSII.2019.2928138.
- [18] 吴庆达, 何书专, 潘红兵, 等. 32位定浮点数正余弦函数FPGA实现方法[J]. 微电子学与计算机, 2012, 29(1): 113-116. DOI: 10.19304/j.cnki.issn1000-7180.2012.01.027.
- [19] 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. DOI: 10.19304/j.cnki.issn1000-7180.2018.12.007.
- [20] 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. DOI: 10.1109/TCSII.2017.2732451.