

基于逐级位宽递增优化的高效流水线 CORDIC 硬件设计

马 昕^{1,2}, 刘 胜^{1,2}, 吴振宇^{1,2*}, 常俊胜^{1,2}, 原 博^{1,2}

(1. 国防科技大学计算机学院, 湖南长沙 410073; 2. 先进微处理器芯片与系统重点实验室(国防科技大学), 湖南长沙 410073)

摘 要: 坐标旋转数字计算(CoOrdinate Rotation DIgital Computer, CORDIC)算法利用移位和加减运算实现多类函数计算, 具有结构规则、易于流水化等特点, 广泛应用于数字信号处理、通信和嵌入式系统等领域。但随着应用规模与计算精度的提高, 系统设计面临兼顾高吞吐率与低硬件开销的严峻挑战。传统流水线 CORDIC 实现通常采用预缩放初值, 并在各迭代级配置统一最大位宽, 导致迭代初期尚未产生有效低位数据时仍实例化全字长硬件, 存在显著的硬件开销。针对这一问题, 本文提出一种基于逐级位宽递增优化与重定时技术的高效流水线 CORDIC 架构。首先, 采用整数初始化并将固定缩放因子补偿外置或后置的方法, 解除预缩放方式对数据通路的全字长约束, 在此基础上, 通过分析移位加减过程中低位有效信息的生成特征, 建立逐级位宽递增模型。理论分析表明, 在达到目标位宽前, 数据有效位宽随迭代次数呈二次增长, 据此按需裁剪前序迭代级数据通路的寄存器和移位加法器, 使迭代初期仅实例化覆盖当前有效信息的逻辑资源, 在保持基本微旋形式和计算精度不变的条件下降低硬件开销。其次, 针对非均匀位宽导致的级间延迟不平衡问题, 对数据通路关键路径进行分析, 并利用前序低位宽流水级的时序裕量, 引入低位宽流水级合并与重定时优化, 进一步减少流水寄存器及相关面积、功耗开销。实验结果表明, 在 8 位有符号定点数据格式下, 与统一位宽 CORDIC 相比, 该架构使电路面积减少 16.72%, 逻辑单元数量减少 22.23%, 功耗降低 13.97%, 结合流水级合并与重定时后, 相比未重定时的统一位宽 CORDIC, 面积减少 33.2%, 功耗降低 20%。该架构在不同精度配置下均保持显著优势, 具有良好的可扩展性。该方法还可扩展至采用整数或低精度常数初始化、保持移位加减迭代形式且缩放补偿可外置的其他 CORDIC 变体, 为高吞吐、资源受限的数字信号处理系统提供低面积、低功耗的 CORDIC 硬件实现方案。

关键词: CORDIC; 流水线; 位宽优化; 重定时; 面积优化; 低功耗设计

基金项目: 国家“万人计划”青年拔尖人才支持计划(No.ZC514C0882503)

中图分类号: TN492; TP302

文献标识码: A

文章编号: 0372-2112(XXXX)XX-0001-14

电子学报 URL: <http://www.ejournal.org.cn>

DOI: 10.12263/DZXB.20260320

Efficient Pipelined CORDIC Hardware Design Based on Stepwise Word-Length Increment Optimization

MA Xin^{1,2}, LIU Sheng^{1,2}, WU Zhenyu^{1,2*}, CHANG Junsheng^{1,2}, YUAN Bo^{1,2}

(1. College of Computer Science and Technology, National University of Defense Technology, Changsha, Hunan 410073, China;

2. Key Laboratory of Advanced Microprocessor Chips and Systems (National University of Defense Technology), Changsha, Hunan 410073, China)

Abstract: Coordinate rotation digital computer (CORDIC) algorithms implement a wide range of function evaluations using shift and add/subtract operations. Owing to their regular structures and natural suitability for pipelining, CORDIC architectures have been widely used in digital signal processing, communication, and embedded systems. However, with the increasing application scale and computational precision, achieving both high throughput and low hardware overhead has become increasingly challenging. Conventional pipelined CORDIC implementations generally employ prescaled initial values and a uniform maximum word length across all iteration stages. As a result, full-word-length hardware is still instantiated in the early iterations even though valid low-order data have not yet been generated, leading to considerable redundant hardware overhead. To address this problem, this paper proposes an efficient pipelined CORDIC architecture based on stepwise word-length increment optimization and retiming. First, integer initialization is adopted, while compensation for the fixed scaling factor is moved outside the core datapath or deferred to a post-processing stage, thereby removing the full-word-length constraint imposed by prescaling. On this basis, the generation characteristics of

valid low-order information during shift-add iterations are analyzed, and a stepwise word-length increment model is established. Theoretical analysis shows that, before the target word length is reached, the effective word length increases quadratically with the iteration index. According to this model, the registers and shift-adders in the X and Y datapaths of the early iteration stages are trimmed on demand, so that only the logic resources required to represent the currently effective information are instantiated in the early iterations. In this way, hardware overhead is reduced while preserving the basic micro-rotation form and computational accuracy of the original CORDIC algorithm. Second, to address the inter-stage delay imbalance caused by the nonuniform word-length distribution, the critical paths of the datapath are analyzed. Since the early low-word-length pipeline stages exhibit shorter carry-propagation paths and therefore larger timing slack, low-word-length pipeline stage merging and retiming optimization are introduced to exploit this slack. By merging adjacent early pipeline stages and redistributing pipeline registers through retiming, the proposed architecture further reduces pipeline-register overhead as well as the associated area and power consumption. Experimental results show that, under an 8-bit signed fixed-point configuration, the proposed stepwise word-length increment architecture reduces circuit area by 16.72%, the total number of logic cells by 22.23%, and power consumption by 13.97% compared with the uniform-word-length CORDIC. With low-word-length pipeline stage merging and retiming, the area and power consumption are further reduced by 33.2% and 20%, respectively, compared with the unretimed uniform-word-length CORDIC. The proposed architecture also maintains significant area advantages under different precision configurations, demonstrating good scalability. In addition, the proposed method can be extended to other CORDIC variants that adopt integer or low-precision constant initialization, retain shift-add-based iterative updates, and allow scaling-factor compensation to be performed outside the core datapath. Therefore, the proposed architecture provides a low-area and low-power CORDIC hardware implementation for high-throughput and resource-constrained digital signal processing systems.

Keywords: CORDIC; pipelining; word-length optimization; retiming; area optimization; low-power design

Foundation Item(s): The Fourth Cohort Funding for High-Level Talent Cultivation Program (No.ZC514C0882503)

0 引言

坐标旋转数字计算 (CoOrdinate Rotation DIgital Computer, CORDIC)^[1-2] 算法以简单的移位与加减法代替复杂的乘法来实现多类初等与超越函数, 在数字信号处理^[3]、通信与嵌入式系统^[4]等场景得到广泛应用。随着应用规模与精度需求持续提升, 系统设计面临着既要满足高吞吐, 又要兼顾面积与功耗的挑战, 两者之间的矛盾日益突出。尽管 CORDIC 线性速率收敛, 但仍需要大量迭代来保持足够的输出精度^[5]。因此, 运算速度始终受限于迭代次数和时钟周期。

为了突破上述性能瓶颈, 现有的 CORDIC 优化研究主要沿两个方面展开。在架构方面, 并行 (parallel)^[6] 和流水线 (pipelining)^[7] 是提升性能的常用手段, 并行 CORDIC 是根据输入角度的二进制表示直接预测出每次基本迭代的旋转方向以消除数据相关^[6], 但通常会带来显著的硬件开销, 流水线 CORDIC 将各次迭代展开为多级硬件单元, 能够显著提高吞吐率, 因此更适合高速数字信号处理场景。在此基础上, Meher 等人^[8] 通过精确的比特级关键路径分析, 提出了非均匀插入流水线寄存器的策略以平衡级间延迟。

在算法方面的 CORDIC 优化主要包括高基数 (High-Radix)、旋转方向优化和缩放因子简化等。高基数方法通过增大单次旋转步长或组合多种基数旋

转来加快收敛速度。Chen 等人^[9] 提出高基数广义双曲 CORDIC (High-Radix Generalized Hyperbolic CORDIC, HGH-CORDIC), 将高基数思想扩展至广义双曲坐标系, 可用于任意固定底数指数和对数函数计算, 并进一步给出了对应的硬件实现^[10]。旋转方向优化方法通过角度选择或旋转序列重构减少方向判决和冗余旋转, Shen 等人^[11] 提出面向循环神经网络激活函数的 CORDIC 架构, 采用贪心角度选择策略确定每级旋转角, 使迭代过程保持固定旋转方向, 从而降低旋转控制复杂度。缩放因子简化方法则通过独立补偿通路、免缩放 (scaling-free) 或后处理补偿降低缩放因子对核心迭代路径的影响, Paz 等人^[12] 提出的面向反正弦和反余弦计算的 CORDIC 方法将微旋转增益补偿从坐标迭代通路中分离出来, 构造独立的补偿通路, 并通过移位加法迭代更新, 从而避免实数乘法, 降低了双迭代方法的单级硬件复杂度。Shen 等人^[11] 则在后处理阶段利用函数关系抵消非恒定缩放因子, 避免在迭代过程中实时更新缩放因子。

上述研究主要从迭代次数、旋转步长、旋转方向和缩放因子处理方式等算法层面提升 CORDIC 性能, 但对展开流水线中各级 X 、 Y 数据通路的实际有效位宽需求关注相对较少。在现代通信和信号处理系统中, CORDIC 常作为数字下变频 (Digital Down Conversion, DDC)^[13] 或 FFT 处理器^[3] 的子模块使用, 其固定缩放因

子可与系统后端乘法、归一化或量化环节合并处理^[14]。在该类应用中,若CORDIC核采用整数初始化,并将缩放因子补偿外置或后置,则前序迭代级的低位有效信息并非初始时全部存在,而是随移位加减过程逐级生成。若各级寄存器和移位加法器仍按统一目标位宽实现,则迭代初期会产生冗余位宽开销。

针对上述问题,本文提出一种基于逐级位宽递增优化的高效流水线CORDIC架构。该方法采用整数初始化和外置缩放补偿策略,打破传统预缩放方式带来的全字长约束;通过分析各级迭代的数据动态范围,建立逐级位宽递增模型,对前序流水级的寄存器和移位加法器进行按需裁剪;同时,针对非均匀位宽导致的级间延迟不平衡问题,引入低位宽流水级合并与重定时优化,将前序级时序裕量进一步转化为面积收益。

本文的主要贡献如下:

(1)提出一种面向展开流水线CORDIC的逐级位宽递增模型,基于整数初始化条件下低位有效信息随迭代逐步生成的特征,确定各级 X 、 Y 数据通路所需的最小有效位宽;

(2)基于该模型设计逐级位宽递增流水线架构,对前序级寄存器和移位加法器进行按需裁剪,在保持计算精度和基本迭代形式不变的前提下降低面积与功耗;

(3)针对非均匀位宽导致的级间时序不平衡问题,引入低位宽流水级合并与重定时优化,将前序级时序裕量转化为进一步的寄存器和面积收益。

1 相关背景

1.1 CORDIC算法

CORDIC算法^[1]由Volder于1959年提出,是一种通过迭代方式实现多种数学运算(如三角函数、双曲函数、乘法、除法、平方根等)的算法。CORDIC算法工作在圆周、双曲和直线三种不同的坐标系下,通过一系列预定的旋转角,可以实现高效的向量旋转与计算。其工作原理为逐步逼近目标角度,并通过一系列加法和移位操作实现迭代收敛。在圆周坐标系下的旋转可以由式(1)描述:

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} \cos \theta_n & -\sin \theta_n \\ \sin \theta_n & \cos \theta_n \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \quad (1)$$

表示第 n 次迭代下的坐标集合, n 为从0开始的整数, θ_n 为第 n 次迭代的旋转角,从式(1)中提取 $\cos \theta_n$,可以得到式(2):

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos \theta_n \begin{bmatrix} 1 & -\tan \theta_n \\ \tan \theta_n & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \quad (2)$$

θ_n 的选取应使下一次迭代的坐标集合

(X_{n+1}, Y_{n+1}) 的值易于计算,这里将 $\tan \theta_n$ 简化为 2^{-n} ,即 $\theta_n = \arctan 2^{-n}$,则式(2)可以转化为

$$\begin{bmatrix} X_n \\ Y_n \end{bmatrix} = \prod_{i=0}^{n-1} \cos \theta_i \begin{bmatrix} 1 & -s_i 2^{-i} \\ s_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} \quad (3)$$

s_i 为1或-1表示第 i 次迭代旋转角的方向,其正负取值依赖于第 n 次迭代的总旋转角 Z_n ,在旋转模式下目标旋转角 θ 是给定的,设 $Z_0 = \theta$,则 Z_n 的取值为 θ 与已迭代角度的差值,即 $\theta - \sum_{i=0}^{n-1} s_i \theta_i$ 。

Walther^[2]通过引入一个新的参数 m 将CORDIC的三种坐标系统(圆周、双曲、直线)和两种工作模式(旋转和向量)推广并统一到一个表达式中,如式(4)所示

$$\begin{aligned} X_{n+1} &= X_n - m s_n (2^{-n} Y_n), \\ Y_{n+1} &= Y_n + s_n (2^{-n} X_n), \\ Z_{n+1} &= Z_n - s_n \theta_n \end{aligned} \quad (4)$$

其中, $m = \{1, -1, 0\}$ 分别对应于圆周、双曲和直线坐标系,对应的 θ_n 值分别为 $\arctan 2^{-n}$ 、 $\operatorname{arctanh} 2^{-n}$ 和 2^{-n} ,对于第 n 次迭代,如果 $Z_n \geq 0$ 则 s_n 为1,如果 $Z_n < 0$ 则 s_n 为-1。标准CORDIC算法在旋转过程中会引入一个增益,即为式(3)中 $\prod_{i=0}^{n-1} \cos \theta_i$ 的倒数,将其记为缩放因子

K ,在双曲旋转模式和圆周坐标系下缩放因子 K 固定,双曲向量模式下的CORDIC可以忽略缩放因子 K ,因为收敛后只需要 Z 坐标的值,在直线坐标系下缩放因子 K 的值为1,也可以忽略。式(4)未显式执行缩放补偿,因此其输出坐标包含缩放因子 K ,表1列出了使用式(4)统一的CORDIC表达式可计算的函数。

1.2 CORDIC硬件实现方式

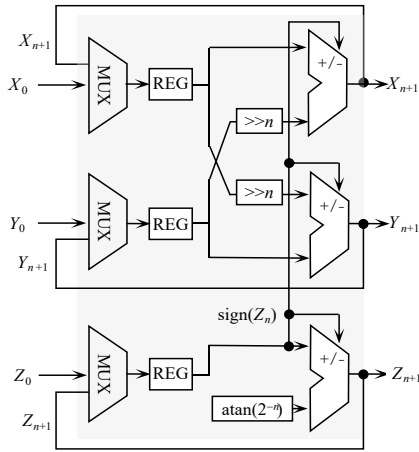
实现CORDIC算法主要有迭代法和流水线法^[15],其对应的硬件结构如图1所示。如图1(a)所示,迭代实现的CORDIC是通过在单个功能单元上进行时间复用来计算的,虽然复用运算单元节约了大量面积,但其每次迭代需处理不同的位移量,必须引入复杂的桶形移位器,且串行处理方式导致数据吞吐量较低,限制了其在高速系统中的应用。

相比之下,流水线CORDIC实现如图1(b)所示,该实现方式将迭代过程展开为级联的多级硬件单元,级间的方框代表流水线寄存器,用于隔离各级硬件的组合逻辑并暂存中间结果 X_n 、 Y_n 和 Z_n 。尽管流水线结构以较大的面积开销换取高吞吐率,但其各级移位可通过硬连线实现,无需桶形移位器,因而能够缩短关键路径延迟。与此同时,多级流水单元可以并行处理不同输入数据,从而提高系统吞吐率,显著提升性能。通常,流水线的级数与迭代次数保持一致。

表 1 使用CORDIC算法可计算的函数列表

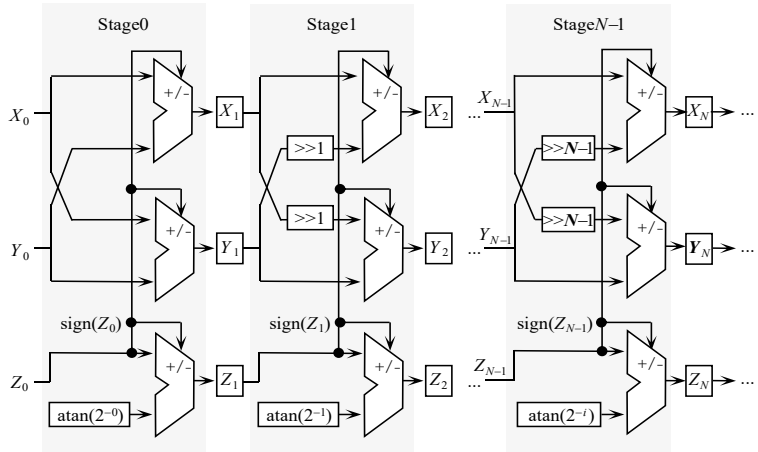
Table 1 Evaluation of different functions using CORDIC algorithm

坐标系	旋转		向量	
	输出	可计算函数	输出	可计算函数
圆周	$X_n = K (X_0 \cos Z_0 - Y_0 \sin Z_0)$ $Y_n = K (Y_0 \cos Z_0 + X_0 \sin Z_0)$ $Z_N \approx 0$	三角函数: $\cos \theta = X_n$ $\sin \theta = Y_n$	$X_n = K \sqrt{(X_0)^2 + (Y_0)^2}$ $Y_n \approx 0$ $Z_N = Z_0 + \tan^{-1}(Y_0/X_0)$	反正切函数: $Z_N = \tan^{-1}(Y_0/X_0)$
线性	$X_n = X_0$ $Y_n = Y_0 + X_0 Z_0$ $Z_n \approx 0$	乘法: $Y_n = X_0 Z_0$	$X_n = X_0$ $Y_n \approx 0$ $Z_n = Z_0 + Y_0/X_0$	除法: $Z_n = Y_0/X_0$
双曲	$X_n = K (X_0 \cosh Z_0 - Y_0 \sinh Z_0)$ $Y_n = K (Y_0 \cosh Z_0 + X_0 \sinh Z_0)$ $Z_N \approx 0$	指数函数: $e^\theta = Y_N$ $= (\cosh \theta + \sinh \theta)$	$X_n = K \sqrt{(X_0)^2 - (Y_0)^2}$ $Y_n \approx 0$ $Z_N = Z_0 + \tanh^{-1}(Y_0/X_0)$	反双曲函数: $Z_N = \tanh^{-1}(Y_0/X_0)$



(a) 迭代实现

(a) Iterative implementation



(b) 流水线实现

(b) Pipelined implementation

图 1 两种不同实现的CORDIC硬件结构

Figure 1 CORDIC hardware architectures with two different implementations

2 逐级位宽递增设计

2.1 动机

在高吞吐场景中, CORDIC算法常常被映射到流水线架构中以实现高时钟频率和高吞吐。此时, 数据通路位宽和迭代级数是影响CORDIC面积与功耗的主要因素。基于图1(b)的流水线CORDIC架构, 图2展示了在8位数据位宽下, 前4级流水线的比特级电路实现细节(略去Z通路), 其中 X_n 和 Y_n 分别代表 X_n 和 Y_n 的第 n 位, X 与 Y 通路交叉连线以更直观地呈现硬连线的移位逻辑。 $\text{sign}Z_n$ 表示第 n 次迭代Z通路的符号位, 对应式(4)中的 s_n , 通过驱动各级输入端的异或门阵列与全加器进位端, 实现加减法操作的控制。

设在圆周坐标系旋转模式下求解目标角的正余

弦值。图2(a)为传统的统一最大位宽架构, 该架构普遍采用全字长寄存器配置, 这是为了抵消CORDIC迭代产生的缩放因子 K 。具体来说, 该架构通常将初始坐标 (X_0, Y_0) 设置为缩放因子的倒数 $(1/K, 0)$ 。由于 $1/K$ (约等于0.607 2)为无限不循环小数, 为了防止量化误差在迭代初期的累积, 硬件必须从第0级迭代开始即实例化足以覆盖最终目标精度的8位全字长寄存器与全加器。这种策略导致了各级硬件规模受限於统一最大位宽约束, 无法适配数据有效性逐级变化的特征, 因此, CORDIC算法仍存在进一步的位宽优化空间。

相比之下, 若打破预缩放的全字长约束, 使用整数坐标 $(1, 0)$ 作为初始坐标输入, 如图2(b)所示的逐级位宽递增架构。在迭代初期, 由于右移位有限,

X 与 Y 通路数据的低位部分尚未生成有效信息,随着右移位数的增加,有效位宽呈现逐级递增的特性,故硬件仅需实例化能够覆盖当前有效位宽的寄存器和加法器。图2(b)中,前4级流水线的有效位宽分别为2、3、5和8,相比于统一的8位位宽设计,可显著降低面积开销,红色虚线区域即为面积节省区域。这种按需分配的策略可使得迭代初期的硬件规模呈阶梯形分布特征,在完整保留数据通路中所有有效信息位的同时,最大程度地减少了逻辑门与触发器的资源消耗。

由上所述,本文提出的逐级位宽递增设计,其带来的面积优化主要来自以下两个方面:

(1) 时序逻辑的直接缩减

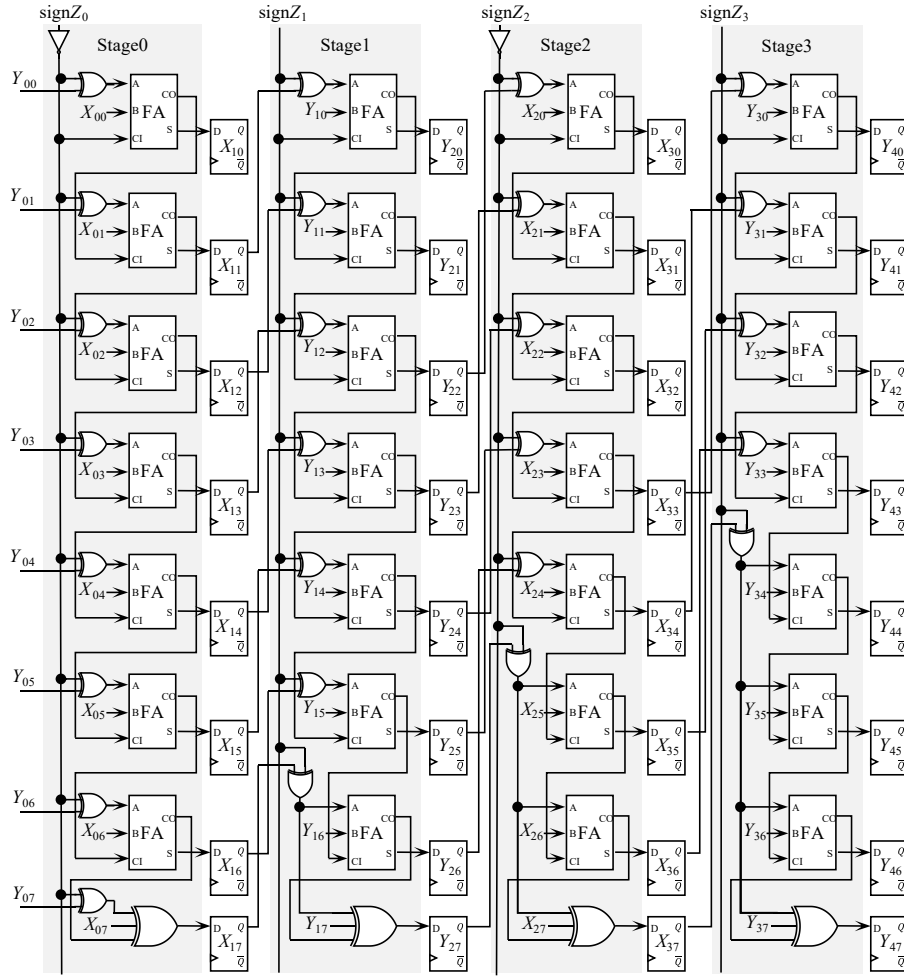
根据CORDIC各级迭代的实际有效位宽需求,直

接减少了 X 、 Y 通路流水线寄存器的实例化数量。

(2) 组合逻辑的同步缩减

各级迭代中加法器的位宽由其前级寄存器的位宽决定, X 、 Y 寄存器数量的减少将同步减少其关联加法器的实例化位宽。

虽然采用整数初始化会导致输出结果中包含缩放因子 K ,但在实际的硬件系统中(如DDC或FFT处理器),这一常数乘法通常可以被合并到系统后端的归一化(normalization)或量化(quantization)步骤中统一处理,无需在CORDIC核心中单独设置补偿电路,这样不仅避免了在迭代路径上引入额外的乘法器开销,还能够使用位宽递增模型进一步优化CORDIC核心的面积。



(a) 传统统一位宽架构

(a) Conventional uniform word length architecture

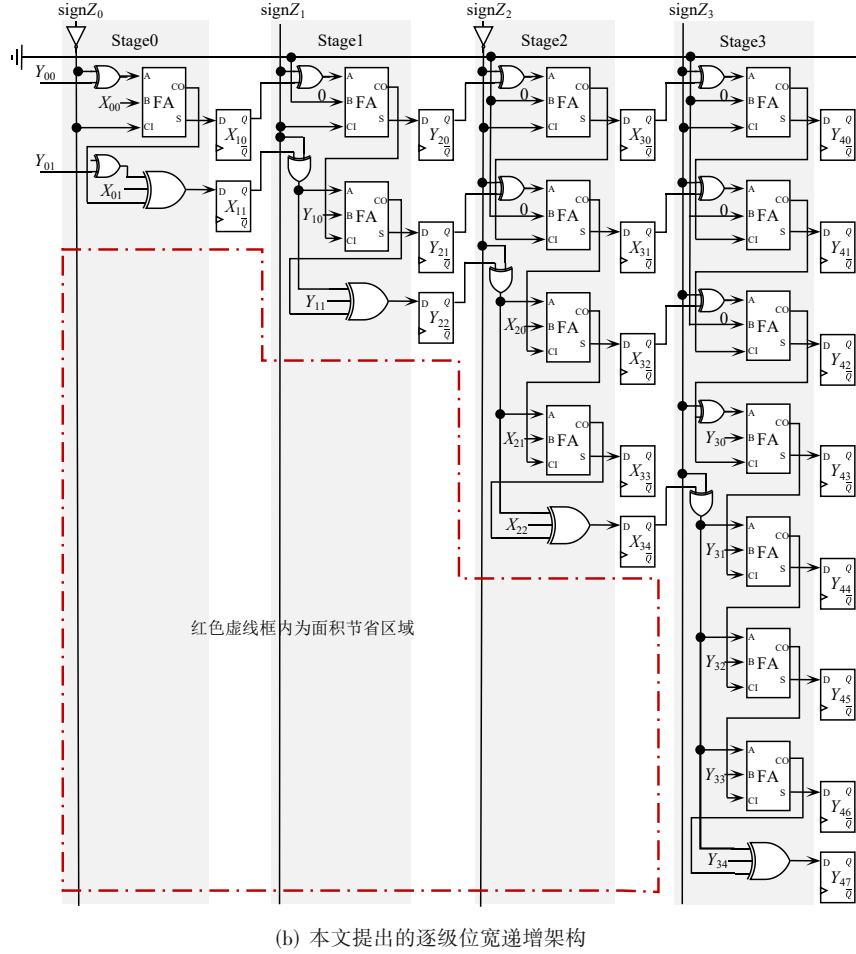


图2 8位字长下前4级流水线CORDIC的比特级电路实现

Figure 2 Bit-level circuit implementation of the first four pipeline stages of an 8-bit CORDIC

2.2 位宽递增模型

在CORDIC算法的迭代过程中,数据位宽的需求并非从一开始就是恒定的。CORDIC算法的运算过程如表2所示,设初始坐标 (X_0, Y_0) 设置为 $(1, 0)$,在Stage0中, X_0, Y_0 未进行右移,此时有效小数位宽 $F=0$ 。在Stage1中, X_1, Y_1 分别右移1位后引入了

第1位有效小数,权重为 2^{-1} ,此时有效小数位宽 $F=1$ 。在Stage2中,上一级输出的带有一位小数的 X_2, Y_2 在运算时进一步右移2位。原本权重为 2^{-1} 的小数被移至权重 2^{-3} 的位置。此时,有效小数位宽 $F=1+2=3$ 位。同理,在Stage3中,有效小数位宽 $F=1+2+3=6$ 位。

表2 CORDIC迭代过程中的坐标数据与有效位宽变化

Table 2 Coordinate data and effective word-length variation during CORDIC iterations

迭代方程	初始化	Stage0($F=0$)	Stage1($F=1$)	Stage2($F=3$)	Stage3($F=6$)
$X_{n+1} = X_n \mp (2^{-n} Y_n)$	$X_0=0b01.$	$X_1=0b01.-0b00.$ $=0b01.$	$X_2=0b01.+0b00.\overset{!}{1}$ $=0b01.1 \xrightarrow{Y_1 \text{ 右移 1 bit}}$	$X_3=0b01.1-0b00.0\overset{!}{0}\overset{!}{1}$ $=0b01.011 \xrightarrow{Y_2 \text{ 右移 2 bit}}$	$X_4=0b01.011+0b00.000\overset{!}{1}\overset{!}{1}\overset{!}{1}$ $=0b01.011111 \xrightarrow{Y_3 \text{ 右移 3 bit}}$
$Y_{n+1} = Y_n \pm (2^{-n} X_n)$	$Y_0=0b00.$	$Y_1=0b00.+0b01.$ $=0b01.$	$Y_2=0b01.-0b00.\overset{!}{1}$ $=0b00.1 \xrightarrow{X_1 \text{ 右移 1 bit}}$	$Y_3=0b00.1+0b00.0\overset{!}{1}\overset{!}{1}$ $=0b00.111 \xrightarrow{X_2 \text{ 右移 2 bit}}$	$Y_4=0b00.111-0b00.001\overset{!}{0}\overset{!}{1}\overset{!}{1}$ $=0b00.101101 \xrightarrow{X_3 \text{ 右移 3 bit}}$
$Z_{n+1} = Z_n - s_n \arctan 2^{-n}$	$Z_0=00.010000$	$Z_1=0b00.010000-$ $0b00.100000$ $=0b11.110000$	$Z_2=0b11.110000+$ $0b00.010011$ $=0b00.000011$	$Z_3=0b00.000011-$ $0b00.000111$ $=0b11.111100$	$Z_4=0b11.111100+$ $0b00.000100$ $=0b00.000000$

由此可见,第 n 次迭代不仅传递了上一次迭代的精度,还通过右移 n 位引入了新的 n 位有效位宽,如表2中红色虚线后方的比特位。这种运算规律使得数据位宽随着迭代的推进呈现出逐级递增的特性。本文基于此特性提出位宽递增模型,定义 F_n 为第 n 次迭代需保留的小数位宽。理论上 F_n 应为从Stage1到Stage n 移位量的总和,如式(5)所示:

$$F_n = \sum_{j=1}^n j = n(n+1)/2 \quad (5)$$

其中, $n=0,1,\dots,N-1$,表示当前的迭代次数。 F_n 随 n 二次增长,在加入一位符号位和一位整数位后,第 n 次迭代的有效总位宽 $W_n = F_n + 2$ 。 W_n 随迭代次数 n 的变化趋势如图3所示。

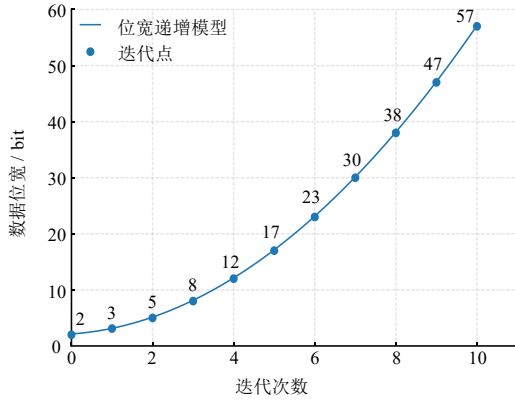


图3 基于位宽递增模型的位宽变化趋势

Figure 3 Word-length variation based on the stepwise increment model

然而,在实际硬件系统中,数据位宽不可能无限增长,而是受限于系统初始设置的目标位宽 W_{target} ,这与目标精度有关。当积累的有效总位宽 W_n 超过目标位宽 W_{target} 后,更低位的数据将被视为无效从而被截断,此时位宽保持恒定。因此 W_n 可由式(6)精确描述:

$$W_n = \min(W_{\text{target}}, F_n + 2) \quad (6)$$

2.3 数据预处理与数据格式

在圆周坐标系下,标准CORDIC算法的收敛区间约为 $[-99.827^\circ, 99.827^\circ]$ ^[16],无法直接覆盖全圆周 $[0, 2\pi)$ 的输入范围。为此,通常采用四分象限映射的方式对输入的初始角度进行预处理,利用三角函数的对称性,将任意输入角度统一映射至 $[0, \pi/2]$ 的区间内,再进入迭代运算。

针对本文设计,若初始坐标 (X_0, Y_0) 设定为 $(1, 0)$,由于映射后的角度位于第一象限,第0次迭代的旋转方向恒为正,即 $s_0 = +1$,根据式(4)计算可知,第0次迭代后的坐标 (X_1, Y_1) 恒定为 $(1, 1)$,角度为 $Z_1 = Z_0 -$

$\arctan(2^0)$ 。基于以上分析,本文对数据的预处理方式为直接设定坐标 (X_1, Y_1) 以及角度 Z_1 作为流水线的初始输入,如图4所示,使得流水线可以直接从第1次迭代开始实例化,有效减少了一级迭代逻辑,降低了电路面积与延迟。

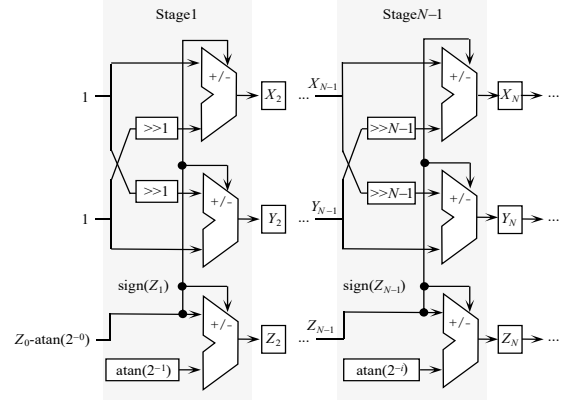


图4 整数初始化的改进CORDIC流水线结构

Figure 4 Improved CORDIC pipeline architecture with integer initialization

对于各数据通路,本文采用有符号定点格式,使用最高有效位(Most Significant Bit, MSB)作为符号位,一位表示整数部分,六位表示小数部分,如图5所示。将这种配置称为 $\langle 8, 6 \rangle$,代表 $\langle$ 总位数,小数部分位数 $\rangle$ 。由于最终输出的 X, Y 的最大值为 K (约等于1.6467),因此一位整数足以确保无溢出。小数部分的位数可以根据实际应用中不同的精度要求进行调整。本文主要讨论定点CORDIC核心的硬件实现,不同应用中的输入量化策略可在系统前端完成。

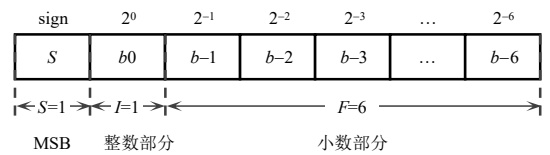


图5 一个8位有符号定点数据格式示意图

Figure 5 Illustration of an 8-bit signed fixed point data format

Z 数据通路采用8位均匀量化格式,将圆周 $[-\pi, \pi]$ 线性映射到8位整数域,简化了在进行象限映射时模 $\pi/2$ 的运算逻辑。

2.4 级间数据对齐

与传统的流水线CORDIC架构不同,由于本设计采用了逐级位宽递增的架构,导致相邻流水级之间的位宽出现不一致,例如从Stage1的3位增至Stage2的5位。

为解决这一问题,对于高位部分,电路采用符号扩展方式进行填充,即复制前一级输出的MSB符号

位至当前级新增高位,以保证不同位宽流水级之间的补码数值一致性。对于低位部分,进行零填充以实现对齐,如图6中红色虚线所示,我们将操作数低位的数据空位接地,一方面可以对齐数据,另一方面可以使低位的全加器逻辑因恒零输入而被化简为低延迟的直连或半加逻辑,从而有效缩短了进位传播链的实际长度,进一步提升了电路的时序性能。

该硬件架构通过RTL级约束,显式描述各级有效数据范围,有利于综合工具消除无效逻辑,从而降低冗余寄存器与组合逻辑开销。

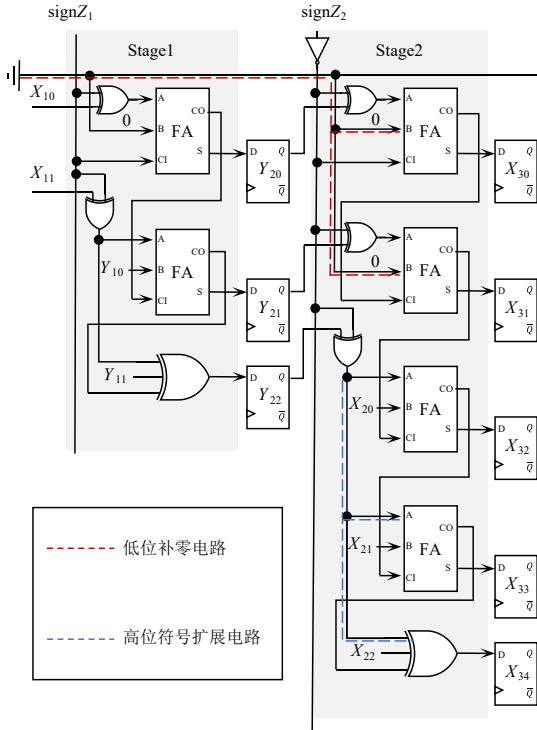


图6 流水级间数据位宽对齐电路

Figure 6 Data word-length alignment circuit between pipeline stages

3 级间时序平衡与重定时

3.1 时序分析

在传统的统一位宽CORDIC架构中,由于每一级的硬件结构高度相似,各级组合逻辑的延迟均由全精度加法器的进位链长度决定,因此每一级的关键路径延迟基本一致,使得各级流水线的时间开销分布均匀。相比之下,逐级位宽递增架构各级数据通路位宽随迭代级次逐步增加,使加法器的进位传播路径相应延长,组合逻辑延迟逐渐增大,进而造成流水线各级延迟分布不均匀。以<8,6>配置为例,其前三级流水线的关键路径如图7所示,迭代初期(如Stage1和Stage2)由于数据位宽较窄,加法器进位传播链较短,关键路径延迟显著低于后续最大位宽的迭代(如

Stage3)。这种物理特性的差异导致级间延迟呈现出显著的非均匀分布,即迭代初期存在较大的时序裕量(timing slack)。

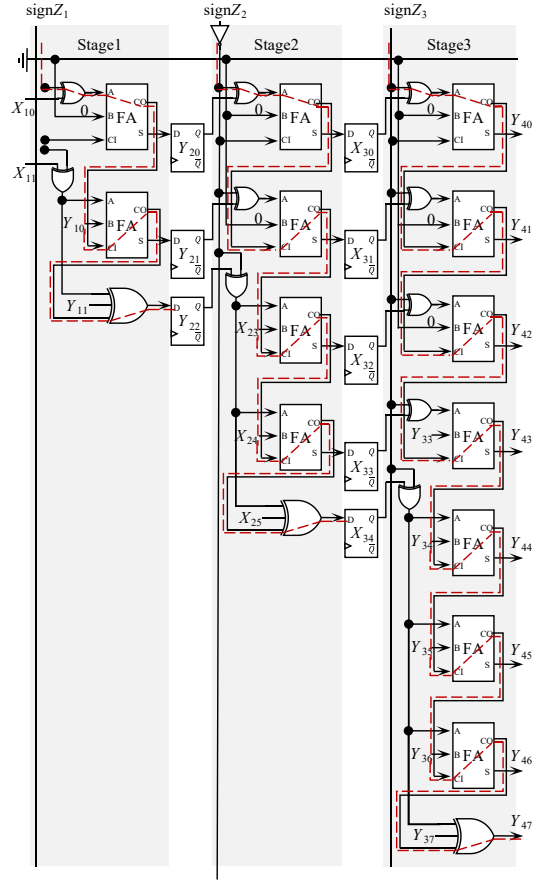


图7 <8,6>配置下前三级流水线关键路径

Figure 7 Critical paths of the first three pipeline stages under the <8,6> configuration

为量化这一特性,本文对基于行波进位加法器(Ripple Carry Adder, RCA)的位宽递增CORDIC数据通路的关键路径进行建模分析。单级流水线的关键路径主要由进位链的传播延迟决定。对于位宽为 W 的流水级,其传播延迟 $T_{\text{Stage}}(W)$ 可表示为

$$T_{\text{Stage}}(W) \approx T_{\text{INV}} + T_{\text{XOR}} + T_{\text{FAAC}} + (W-2)T_{\text{FACC}} + T_{\text{FACS}} \quad (7)$$

其中, T_{INV} 为反相器延迟, T_{XOR} 为异或门延迟, T_{FAAC} 为全加器从输入到进位输出的延迟, T_{FACC} 为全加器从进位输入到进位输出的延迟, T_{FACS} 为全加器从进位输入到和输出的延迟。式(7)表明,各级延迟与位宽 W 呈线性正相关。结合前文的位宽递增模型,当前迭代的位宽 W 会随迭代次数 n 的增加而增长,进而导致加法器进位链的传播延迟随 n 逐级上升。表3列出了在不考虑目标位宽截断的条件下,当迭代次数 $1 \leq n \leq 7$ 时,位宽递增CORDIC数据通路的估算路径延迟。

表3 位宽递增CORDIC数据通路的估算路径延迟

Table 3 Estimated delays of the stepwise word-length increment CORDIC datapath

迭代次数	位宽 W/bit	传播延迟 $T_{\text{stage}}(W)/\text{ns}$
1	3	0.63
2	5	0.79
3	8	1.03
4	12	1.35
5	17	1.75
6	23	2.23
7	30	2.79

其中估算参数设置为 $T_{\text{INV}}=0.10 \text{ ns}$, $T_{\text{XOR}}=0.12 \text{ ns}$, $T_{\text{FAAC}}=0.18 \text{ ns}$, $T_{\text{FACC}}=0.08 \text{ ns}$, $T_{\text{FACS}}=0.15 \text{ ns}$ 。由表中数据可知,随着迭代次数的增加,在未达到最大位宽时,运算所需的有效位宽由3位逐步扩展至30位。受进位链长度增加的影响,单级路径延迟呈现出显著的单调递增趋势。相比 Stage7 的 2.79 ns 传播延迟, Stage1 的延迟仅为 0.63 ns,二者相差 2.16 ns,表明前级流水阶段存在较大的时序裕量,导致各流水级之间产生明显的时序不平衡问题。当数据位宽扩展至最大位宽时,流水级的路径延迟保持稳定,且接近传统统一位宽架构的单级延迟,成为限制系统最高频率的瓶颈。

3.2 低位宽流水级合并与重定时优化

针对前节分析指出的时序不平衡问题,为了充分利用迭代初期的时序裕量,本文引入了重定时技术对流水线进行全局优化。重定时是由 Leiserson 等人^[17]提出的一种经典的电路变换技术,其核心思想是在不改变系统的输入输出特性的前提下,通过改变寄存器在组合逻辑中的位置,来调整电路的关键路径延迟。重定时方法在许多应用中用于减少时钟周期、功耗和寄存器数量^[18]。

CORDIC 算法的展开结构是一个前馈 (feed forward) 的无环图 (Directed Acyclic Graph, DAG)。虽然流水线寄存器通常被插入到单个 CORDIC 迭代级之间,如图 1(b)所示,但它们基本上可以被放置在数据通路中的任何割集 (cut-set) 位置。因此,该架构完全满足应用重定时优化的前提。

在本文提出的逐级位宽递增架构中,迭代初期由于数据位宽较窄,组合逻辑延迟明显低于后续达到最大位宽的迭代延迟,因而存在可利用的时序裕量。基于该特点,本文首先将相邻的前序低位宽迭代级进行流水边界合并,移除其级间寄存器,使多个低位宽微旋转处于同一寄存器到寄存器路径中,然后在逻辑综合阶段结合重定时,对寄存器位置进行重新分布。

在 <8,6> 配置下,本文将 X、Y、Z 通路中的前两级

迭代合并至同一流水级,使路径延迟较低的 Stage1 与 Stage2 移位加法逻辑处于同一流水级中,如图 8 所示,该优化方式不仅减少了一个流水级和 CORDIC 算法迭代次数,还能够在保持流水结构相对稳定的同时降低迭代初期寄存器数量。但由于合并后的单个流水级包含两级低位宽微旋转逻辑,单周期组合逻辑深度增加,因此其最小可达时钟周期大于未合并结构,最高工作频率受到一定限制。

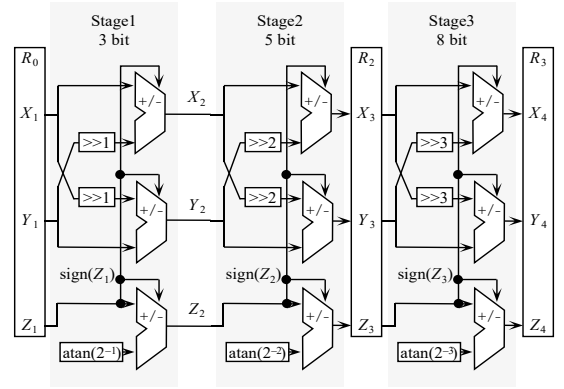


图8 <8,6>配置下低位宽流水级合并与重定时优化示意图

Figure 8 Low-word-length pipeline stage merging and retiming optimization under the <8,6> configuration

4 实验与分析

为验证本文所提出的逐级位宽递增 CORDIC 架构的有效性以及其结合重定时优化后的实际效果,本文从硬件面积、时序性能和功耗开销三个方面进行实验评估。为保证对比公平,实验中的统一位宽基线与本文架构均采用相同的数据预处理与输出恢复逻辑,二者的主要区别仅在于 CORDIC 迭代数据通路 X、Y 的位宽分配策略:统一位宽基线在各级迭代中均采用目标最大位宽,本文架构依据前文提出的位宽递增模型对初期迭代级进行逐级位宽裁剪。在此基础上,本文进一步对传统统一位宽架构、逐级位宽递增架构以及逐级位宽递增结合重定时优化的架构分别进行时钟周期扫描评估性能上限。

实验采用 Verilog 描述电路,基于某商用 FinFET 工艺库,所有设计均在相同工艺库、相同 PVT 条件、相同输入输出约束和相同综合流程下完成逻辑综合,并以综合面积、功耗、最小可达时钟周期作为主要评价指标。

4.1 <8,6>配置固定约束下面积与功耗分析

首先,在 <8,6> 定点配置下(即总位宽 8 位,小数位宽 6 位),比较统一位宽 CORDIC 与位宽递增 CORDIC 的面积、功耗和时序。最差负时序裕量 (Worst Negative Slack, WNS) 和总负时序裕量 (Total

Negative Slack, TNS)作为时序收敛指标, $WNS \geq 0$ 表明最关键路径满足目标时钟约束, $TNS = 0$ 表明设计不存在建立时间违例。

本文首先对统一位宽 CORDIC 与位宽递增 CORDIC 进行时钟周期扫描, 结果表明, 两种架构能实现时序收敛的最小时钟周期均为 0.23 ns, 对应最高工作频率约为 4.34 GHz。因此, 本节选取 4.34 GHz 作为固定时钟约束, 评估位宽递增优化的面积和功耗收益。综合结果如表 4 所示。

表 4 <8,6>配置固定约束下不同 CORDIC 架构的综合结果

Table 4 Synthesis results of different CORDIC architectures under a fixed timing constraint for the <8,6> configuration

架构	面积/ μm^2	逻辑单元总个数				功耗/ mW	WNS/ ns	TNS/ ns
		总计	组合 逻辑	时序 逻辑	其他			
统一位宽 CORDIC	741.78	1 201	1 051	143	7	4.08	0	0
位宽递增 CORDIC	617.73	934	799	129	6	3.51	0	0

实验数据显示, 采用逐级位宽递增架构后, 电路总面积由 $741.78 \mu\text{m}^2$ 降至 $617.73 \mu\text{m}^2$, 减少了 16.72%, 逻辑单元总数减少了 22.23%, 这直接验证了位宽递增模型的有效性: 通过缩减迭代初期的数据位宽, 可以在不损失算法精度的前提下降低硬件成本。功耗方面, 统一位宽 CORDIC 的综合功耗为 4.08 mW, 位宽递增 CORDIC 降至 3.51 mW, 降低 13.97%。其主要原因在于, 前级寄存器数量和加法器位宽的减少降低了触发器开关活动和组合逻辑等效负载, 从而减少了时钟网络和数据通路上的动态功耗。此外, 两种架构在该固定时钟约束下均满足时序要求。统一位宽 CORDIC 和位宽递增 CORDIC 的 WNS、TNS 均接近 0, 说明位宽递增优化并未破坏设计的时序收敛能力。综上, 在相同功能、相同预处理和相同时钟约束条件下, 位宽递增 CORDIC 能够同时降低综合面积、逻辑单元数量和功耗开销, 验证了逐级位宽递增模型在该

定点配置下的有效性。

在减少的单元中, 时序单元减少的个数为 14, 为了探究面积缩减的具体来源, 对 X 数据通路各迭代的时序单元数量进行了细粒度统计, 如表 5 所示, Y 通路的统计数据与 X 通路相同。

表 5 X 通路各迭代时序单元个数

Table 5 Number of sequential cells at each stage in the X datapath

迭代次数	统一位宽 CORDIC	位宽递增 CORDIC	减少个数
1	6	2	4
2	8	5	3
3	8	8	0
4	8	8	0
5	8	8	0

从时序单元的减少数可以看出, 本文提出的逐级位宽递增 CORDIC 设计迭代初期对寄存器有效地进行了缩减, 在 Stage1, 寄存器数量相比统一位宽减少了 4 个, 在 Stage2, 随着有效位宽需求的增加, 寄存器数量增至 5 个, 仍比统一架构减少 3 个。从 Stage3 开始, 有效位宽达到了全字长 8 位, 此后两种架构寄存器数量持平。受综合软件对初始输入数据的常量传播 (constant propagation) 优化影响, Stage1 的实测数据 (统一位宽架构为 6, 本文架构为 2) 均略低于理论位宽值 (统一位宽架构为 8, 本文架构为 3), 但这不影响两者相对趋势的对比。

4.2 低位宽流水级合并与重定时效果分析

基于第 3.1 节的时序分析, 虽然位宽优化后面积有所缩减, 但级间延迟呈现出显著的非均匀分布, 迭代初期存在时序裕量。为此, 进一步引入低位宽流水级合并与重定时优化, 将此裕量转化为寄存器缩减。为评估重定时优化的效果, 本文在 <8, 6> 配置下, 以重定时优化架构可达到的最高工作频率 3.85 GHz 作为统一时钟约束, 对统一位宽 CORDIC 和位宽递增 CORDIC 在重定时前后的硬件开销进行综合对比, 结果如表 6 所示。

表 6 <8,6>配置下重定时优化前后的面积与功耗对比

Table 6 Area and power comparison before and after retiming under the <8,6> configuration

架构	流水级数	面积/ μm^2	功耗/mW	WNS/ns	TNS/ns
统一位宽 CORDIC	无重定时	5	658.52	3.40	0
	重定时	5	655.62	3.39	0
位宽递增 CORDIC	无重定时	5	559.51	2.95	0
	重定时	4	439.86	2.72	0

数据表明, 重定时对统一位宽 CORDIC 的面积改善有限, 而对位宽递增架构表现出更明显的面积与功耗收益。对于统一位宽 CORDIC, 单独采用重定时后

综合面积由 $658.52 \mu\text{m}^2$ 降至 $655.62 \mu\text{m}^2$, 仅降低 0.44%; 功耗由 3.40 mW 降至 3.39 mW, 变化幅度较小。

相比之下,位宽递增CORDIC在重定时优化后表现出更明显的面积与功耗收益,经过低位宽流水级合并与重定时后,流水级数由5级减少至4级,面积相对未重定时的位宽递增架构降低21.38%,功耗降低7.8%。相较于未重定时的统一位宽基线,重定时的位宽递增架构在相同约束下面积降低33.2%,功耗降低20%。

上述结果表明,重定时对位宽递增架构的优化效果明显优于统一位宽架构。其原因在于,位宽递增结构在迭代初期具有较短的移位加法路径和更大的时序裕量,低位宽流水级合并将多个低位宽迭代纳入同一优化窗口,使重定时能够更充分地利用级间时序裕量,从而同步降低寄存器开销、时钟负载和组合逻辑面积。与之相比,统一位宽架构中的各级路径延迟相对

均衡,可利用的级间时序裕量较小,重定时主要体现为寄存器重分布和局部功耗优化,对面积的改善有限。

4.3 最小可达时钟周期与最高工作频率分析

固定时钟约束下的综合结果只能反映特定目标频率下的硬件开销,难以体现不同架构的性能上限。为进一步比较统一位宽CORDIC、位宽递增CORDIC以及重定时位宽递增CORDIC在不同约束强度下的面积与功耗,本文对三种架构进行时钟周期扫描实验,结果如图9所示。图9(a)给出了不同时钟周期约束下的综合面积变化,图9(b)给出了对应的综合功耗变化。横轴为目标时钟周期 T_{clk} ,越靠左表示时钟周期约束越苛刻。对于满足 $WNS \geq 0$ 且 $TNS = 0$ 的结果,记录其面积和功耗;不满足的结果将标记为违例点,不参与有效点比较。

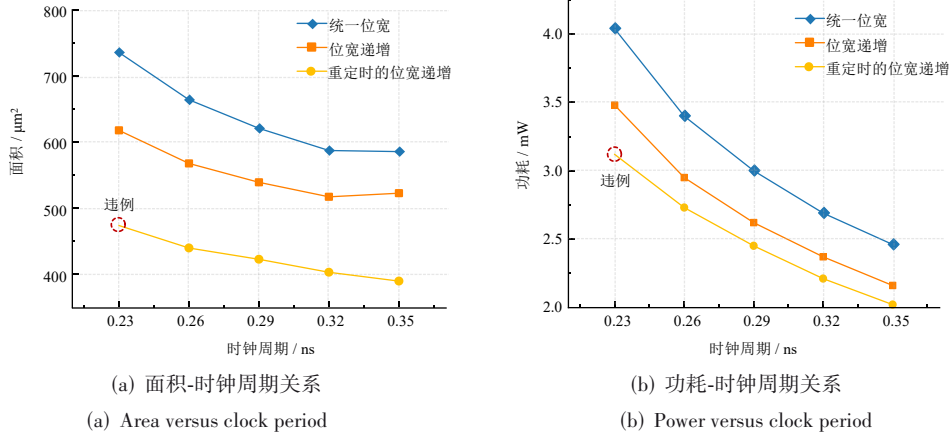


图9 <8,6>配置下不同CORDIC架构的面积/功耗—时钟周期关系图

Figure 9 Area/power versus clock period for different CORDIC architectures under the <8,6> configuration

由图9可知,在 $T_{clk}=0.23$ ns的最苛刻约束下,统一位宽CORDIC与位宽递增CORDIC均能够完成时序收敛,对应最高工作频率约为4.34 GHz。而重定时的位宽递增CORDIC在0.23 ns约束下出现时序违例,其最小可达时钟周期为0.26 ns,对应最高工作频率约3.85 GHz。这说明,逐级位宽递增优化本身并未降低架构的最高工作频率,而重定时的逐级位宽递增优化由于低位宽流水级合并减少了流水级数,增加了单周期组合逻辑深度,其最高可达频率有所下降。

从面积结果看,位宽递增CORDIC在所有有效时钟周期下均低于统一位宽CORDIC。随着时钟周期进一步放宽,三种架构的综合面积整体呈下降趋势,其中重定时的位宽递增CORDIC在满足时序的有效点中面积最小,说明低位宽流水级合并与重定时能够有效降低流水寄存器和组合逻辑开销。

图9(b)显示的功耗变化趋势表明,随着时钟周

期放宽,三种架构的综合功耗均逐渐降低。在相同时钟周期下,位宽递增CORDIC的功耗低于统一位宽CORDIC,而重定时的位宽递增CORDIC在满足时序的约束点上具有最低功耗。其原因在于,位宽递增减少了前序迭代级的寄存器位数和移位加法器位宽,而重定时进一步减少了前序流水边界和相关时钟负载,从而降低了数据通路与时钟网络上的动态功耗。

综上,位宽递增CORDIC在保持与统一位宽CORDIC相同最高工作频率的同时实现了面积和功耗降低,适用于高吞吐场景。重定时的位宽递增CORDIC进一步降低了面积与功耗,但其最小可达时钟周期由0.23 ns增加至0.26 ns,因此更适用于以面积和功耗为优先优化目标的场景。

4.4 扩展至不同配置的硬件开销分析

为验证本文提出的位宽递增模型在不同精度需求下的可扩展性和有效性,实验设置多种数据格式配

置<12, 10>、<22, 20>以及<32, 30>, 分别对应总迭代次数 $N=10$ 、20和30, 进行扩展性测试。在不同配置下, 位宽递增架构相对于统一位宽架构的面积开销数据如表7所示。

表7 不同精度配置下的硬件开销对比

Table 7 Hardware cost comparison under different precision configurations

配置	统一位宽 CORDIC	位宽递增 CORDIC	面积减少/%	时序单元 减少个数
<12,10>	2 313.98	1 923.73	16.86	32
<22,20>	8 676.82	7 824.52	9.82	110
<32,30>	19 673.38	17 585.06	10.48	226

实验数据显示, 随着总迭代次数 N 的增加, 位宽递增架构所减少的时序单元数量呈现明显的上升趋势。这是因为随着目标精度的提高, 有效位宽达到系统最大位宽所需的迭代级数也相应增加, 即有更多的前序流水级能够处于非最大位宽的优化区间内, 从而缩减了更多的电路逻辑。

尽管节省的面积在增加, 但面积节省率从<12, 10>配置的16.86%下降并稳定在10%左右。这表明, 逐级位宽递增方法的优势主要集中在数据有效位宽尚未增长至系统最大位宽的前序阶段。当有效数据填满预设位宽后, 后续所有级数均必须按照预设的最大位宽硬件实现。随着总迭代次数 N 的增加, 最大位宽的流水线级数占比逐渐增大, 构成了电路面积的主体。因此, 尽管迭代初期节省的面积绝对值在增加, 但其相对于庞大的总面积而言, 节省率反而降低。

在以上三种精度配置下, 位宽递增CORDIC架构始终保持了显著的面积优势, 说明逐级位宽递增方法在不同定点精度配置下具有良好的可扩展性, 可为不同资源约束下的CORDIC硬件实现提供参考。

4.5 其他CORDIC算法的对比与适配性分析

本文提出的逐级位宽递增方法本质上是一种微架构级优化方法, 其适用条件并不局限于传统基2圆周坐标系CORDIC。表8列出了近几年先进CORDIC方法与本文方法的对比与适配性分析, 由表8可知, 只要目标CORDIC算法满足以下三个条件, 即可采用类似的位宽递增建模与面积缩减策略:

- (1) X 、 Y 数据通路采用整数或低精度常数初始化, 使得初始数据不含或仅含有限小数有效位;
- (2) 各级迭代仍以移位加减法形式更新中间坐标, 使得低位有效信息随迭代级数逐步生成;
- (3) 缩放因子补偿能够外置或与系统后端归一化、量化环节合并。

对于Paz等人^[12]提出的反正弦和反余弦CORDIC方法, 通过构造独立的 T_i 补偿通路将微旋转增益补

偿分离出来, 其 X 、 Y 、 Z 数据通路仍采用基本移位加减迭代形式, 并采用 $X_0=1$ 、 $Y_0=0$ 、 $Z_0=0$ 的整数初始化。本文方法可用于该算法中 X 、 Y 坐标通路的前序级位宽裁剪, 但 T_i 通路的初值来自外部输入, 不能直接套用本文的整数初始化位宽递增模型, 应单独进行位宽分析。

对于Chen等人^[9]提出的HGH-CORDIC算法, 由于高基数算法缩放因子 K_{hgh} 不再固定, 若在级内进行补偿具有很大的硬件开销, 故在迭代时 X 、 Y 通路采用 $X_0=Y_0=1$ 的整数初始化形式, 而缩放因子 K_{hgh} 在迭代结束后通过查表或Taylor近似等方式进行补偿。因此, 本文方法可作为HGH-CORDIC的补充优化。与基2CORDIC不同的是, 高基数算法每级等效移位深度由具体旋转序列决定, 故其位宽增长规律需要根据实际移位序列重新建立, 不能直接套用本文基2前提下的位宽递增公式。

对于Shen等人^[11]提出的面向RNN激活函数的Greedy-based CORDIC架构, 其双曲旋转路径可在后处理阶段抵消缩放因子, 避免在核心迭代级实时补偿, 故初始输入 X_0 可以是任意常数(文中选择1.5), $Y_0=0$ 。该设计满足上面的三个条件, 可以使用本文的逐级位宽递增方法。由于该设计采用贪心角度选择策略, 每级旋转角和移位深度可能随输入动态变化, 因此若引入本文方法, 需要按照可能出现的最大移位深度建立保守位宽模型。

需要指出的是, 本文方法并不能无条件套用于所有CORDIC变体。对于采用预缩放初始化、迭代级中引入深小数位, 或输入向量本身为全精度外部数据的算法, 此类结构中低位有效信息并不一定随迭代过程逐级生成, 因此不能直接套用本文模型, 而需结合具体数据通路重新建立模型。

5 结束语

本文提出了一种基于逐级位宽递增优化的高效流水线CORDIC硬件设计方案。通过理论建模、架构重构与时序优化, 实现了在保证计算精度前提下的硬件资源压缩。本文的主要工作与结论总结如下:

首先, 本文建立了基于动态范围分析的位宽递增模型, 通过采用外置缩放补偿策略并利用整数初始化特性, 打破了传统架构受限于缩放因子处理而必须采用统一最大位宽的约束。该模型准确描述了流水线各级的数据有效位宽需求, 使得在迭代初期仅需实例化覆盖当前有效信息的最小逻辑资源, 剔除了不必要的组合逻辑与寄存器开销。

其次, 针对非均匀位宽分布导致的级间时序不平衡问题, 本文引入重定时技术对流水线关键路径进行

表8 近几年CORDIC方法的对比与适配性分析

Table 8 Comparison and applicability analysis of recent CORDIC methods

CORDIC方法	计算函数	初始化方式	单级迭代特征	缩放因子处理	迭代次数	与本文方法的适配性
反正弦/反余弦CORDIC ^[12]	反正弦函数 反余弦函数	整数初始化	Radix-2 固定移位	独立T通路近似补偿	$N-1$	按本文中的位宽递增模型对算法进行优化
HGH-CORDIC ^[9]	指数函数、对数函数	旋转模式采用整数初始化	高基数动态旋转系数,等效移位深度随分支变化	可变缩放因子通过查表或近似补偿	$N/\log_2 r$	旋转模式可按最坏移位建立保守位宽模型
Greedy-based CORDIC ^[11]	双曲函数、反正切函数	双曲旋转模式采用低精度常数初始化	固定旋转方向,贪心选择微旋转角	在后处理阶段通过函数关系抵消	$N/2$	双曲旋转通路可建立保守位宽模型
本文	正弦函数、余弦函数	X、Y采用整数常数初始化	保持原有基2微旋转形式	补偿外置或与系统后端融合	$N-2$	微架构级优化,可与满足适用条件的CORDIC算法叠加

注: N 为达到相同目标精度时 Radix-2 CORDIC 所需的迭代次数; r 为高基数 CORDIC 的基数。

了全局优化。该方法有效地将迭代初期存在的显著时序裕量转化为寄存器资源的进一步缩减,实现了时序收敛与面积优化的权衡。

实验结果表明,该架构在不同精度配置下均具有显著的优势。在典型的<8,6>有符号定点数据格式下,与传统统一位宽设计相比,本文提出的架构使电路总面积减少了16.72%,结合重定时优化后,面积降幅进一步达到33.2%。此外,在扩展至<12,10>、<22,20>及<32,30>等更高精度配置的测试中,该设计展现出良好的可扩展性,且在多种数据精度下,均能保持显著的面积节省收益。

此外,本文进一步讨论了所提方法对其他CORDIC算法的适配性。分析表明,逐级位宽递增方法适用于采用低精度初始化、核心迭代保持移位加减结构,且缩放补偿不进入核心迭代的数据通路。对于采用全精度预缩放初始化、前序迭代级已引入深小数位,或输入向量本身来自外部全精度数据的CORDIC算法,不能直接套用本文模型,而需结合具体数据通路重新进行位宽分析。

综上所述,本文提出的基于逐级位宽递增与重定时优化的CORDIC架构,成功实现了硬件资源的高效利用,为资源受限的嵌入式控制系统及高性能数字信号处理应用提供了一种有效的低开销硬件解决方案。

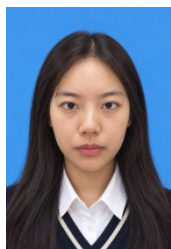
参考文献

- [1] Volder J E. The CORDIC trigonometric computing technique[J]. IRE Transactions on Electronic Computers, 1959, EC-8(3): 330-334.
- [2] Walther J S. A unified algorithm for elementary functions[C]//Proceedings of AFIPS Spring Joint Computer Conference. New York: ACM, 1971: 379-385.

- [3] 雷元武, 陈小文, 彭元喜. DSP 芯片中的高性能 FFT 加速器[J]. 计算机研究与发展, 2016, 53(7): 1438-1446.
Lei Yuanwu, Chen Xiaowen, Peng Yuanxi. A high energy efficiency FFT accelerator on DSP chip[J]. Journal of Computer Research and Development, 2016, 53(7): 1438-1446. (in Chinese)
- [4] 史方显, 曾立, 陈昱, 等. 改进型高速高精度 CORDIC 算法及其在 DDFS 中的应用[J]. 电子学报, 2017, 45(2): 446-451.
Shi Fangxian, Zeng Li, Chen Yu, et al. Direct digital frequency synthesizer based on an improved high speed & high precision CORDIC algorithm[J]. Acta Electronica Sinica, 2017, 45(2): 446-451. (in Chinese)
- [5] Hu Y H. CORDIC-based VLSI architectures for digital signal processing[J]. IEEE Signal Processing Magazine, 1992, 9(3): 16-35.
- [6] 雷元武, 周杰, 葛颖增, 等. 并行 CORDIC 算法的研究及 FPGA 实现[J]. 计算机工程与科学, 2008, 30(8): 75-78.
Lei Yuanwu, Zhou Jie, Ge Yingzeng, et al. Research of the parallel CORDIC algorithm and its implementation in FPGA[J]. Computer Engineering & Science, 2008, 30(8): 75-78. (in Chinese)
- [7] 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.
- [8] Meher P K, Park S Y. Design of cascaded CORDIC based on precise analysis of critical path[J]. Electronics, 2019, 8(4): 382.
- [9] Chen Hui, Quan Lianghua, Liu Weiqiang. HGH-CORDIC:

- A high-radix generalized hyperbolic COordinate rotation digital computer[C]//2024 IEEE 31st Symposium on Computer Arithmetic. Piscataway: IEEE, 2024: 88-95.
- [10] Chen Hui, Quan Lianghua, Chen Ke, et al. High-radix generalized hyperbolic CORDIC and its hardware implementation[J]. IEEE Transactions on Computers, 2025, 74(3): 983-995.
- [11] Shen Wan, Jiang Junye, Li Minghan, et al. Efficient CORDIC-based activation functions for RNN acceleration on FPGAs[J]. IEEE Transactions on Artificial Intelligence, 2025, 6(1): 199-210.
- [12] Paz P, Garrido M. CORDIC-based computation of arcsine and arccosine functions on FPGA[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2023, 70(9): 3684-3688.
- [13] Datta D, Dutta H S, Naskar M K. Heuristic analysis of digital down converter design for software-defined radio applications[J]. Circuits, Systems, and Signal Processing, 2025, 44(1): 684-700.
- [14] Andraka R. A survey of CORDIC algorithms for FPGA based computers[C]//Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays - FPGA '98. New York: ACM, 1998: 191-200.
- [15] 毕卓, 戴益君. 全定制CORDIC运算器设计[J]. 计算机工程与科学, 2011, 33(10): 64-69.
- Bi Zhuo, Dai Yijun. Full custom CORDIC arithmetic unit design[J]. Computer Engineering & Science, 2011, 33(10): 64-69. (in Chinese)
- [16] Hu X, Harber R G, Bass S C. Expanding the range of convergence of the CORDIC algorithm[J]. IEEE Transactions on Computers, 1991, 40(1): 13-21.
- [17] Leiserson C E, Saxe J B. Retiming synchronous circuitry [J]. Algorithmica, 1991, 6(1): 5-35.
- [18] Odugu V K, Rao B J, Babu G H. PAR2DF: Power and area efficient retiming 2D FIR filter architecture using optimized multipliers and adders[J]. Circuits, Systems, and Signal Processing, 2025, 44(12): 9694-9724.

作者简介



马 昕 女, 1999年3月出生于山东省青岛市。现为国防科技大学计算机学院硕士研究生。主要研究方向为微处理器设计与低功耗设计方法。

E-mail: xinma@nudt.edu.cn



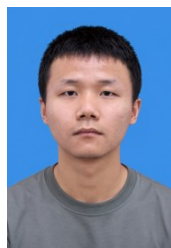
常俊胜 男, 1979年8月出生于河南省南阳市。现为国防科技大学计算机学院研究员。主要研究方向为计算机体系结构、高速互连网络。

E-mail: junshengchang@nudt.edu.cn



刘 胜 男, 1984年1月出生于河南省南阳市。现为国防科技大学计算机学院研究员。主要研究方向为微处理器体系结构和VLSI设计。

E-mail: liusheng83@nudt.edu.cn



原 博 男, 1996年8月出生于陕西省咸阳市。现为国防科技大学计算机学院博士研究生。主要研究方向为微处理器体系结构和VLSI设计。

E-mail: yuanbol8@nudt.edu.cn



吴振宇 男, 1991年7月出生于浙江省衢州市。现为国防科技大学计算机学院助理研究员。主要研究方向为低功耗设计方法学与高效微处理器设计。

E-mail: wuzhenyu@nudt.edu.cn