小 型 微 型 计 算 机 系 统
Journal of Chinese Computer Systems
DOI:10.20009/j.cnki.21-1106/TP.2025-0066
2026 年 2 月 第 2 期
Vol.47 No.2 2026

电路拓扑驱动的晶体管级时序优化算法

成泽祥[1],罗元盛[1],冯超超[2,3],赵振宇[2,3],张曾慧[2,3],成 龙[2,3]

[1](长沙理工大学 计算机学院,长沙 410015)
[2](国防科技大学 计算机学院,长沙 410073)
[3](国防科技大学 先进微处理器芯片与系统重点实验室,长沙 410073)
E-mail:fengchaochao@nudt.edu.cn

摘 要:随着集成电路技术的飞速发展,电路设计的复杂性与日俱增,晶体管级时序优化成为提高电路性能的重要手段.针对由标准单元组成的复杂电路中关键路径延时过大的问题,提出了一种基于电路拓扑结构的晶体管级时序优化算法.该算法通过分析电路拓扑特征,精准识别电流传输的上拉或下拉路径,并结合多种优化策略,对晶体管尺寸进行精细调整,从而有效缩短了关键路径的延时.实验表明,该算法在多种电路场景下均展现出了卓越的时序优化效能.具体而言,在针对路径上所有单元的全面优化策略下,200 条测试路径的前仿测试结果显示,其延时平均降低了 20.7%;而当优化焦点集中于延时敏感单元时,这 200 条路径的前仿测试延时同样实现了 8.1% 的平均降幅.更进一步地,在精选的 10 条路径上,仅对延时敏感单元进行优化并完成版图绘制工作,后仿测试结果表明,这些路径的延时平均减少了 6.8%.这一系列显著的优化成果不仅充分证明了该算法的有效性与实用性,更为未来针对延时敏感单元开展局部全定制电路设计提供了新的思路.

关 键 词:晶体管;电路拓扑结构;电路仿真;时序优化
中图分类号:TN47 **文献标识码**:A **文 章 编 号**:1000-1220(2026)02-0504-09

Topology-driven Transistor-level Timing Optimization Algorithm

CHENG Zexiang[1],LUO Yuansheng[1],FENG Chaochao[2,3],ZHAO Zhenyu[2,3],ZHANG Zenghui[2,3],CHENG Long[2,3]
[1](School of Computer Science and Technology,Changsha University of Science and Technology,Changsha 410015,China)
[2](College of Computer Science and Technology,National University of Defense Technology,Changsha 410073,China)
[3](Key Laboratory of Advanced Microprocessor Chips and Systems,National University of Defense Technology,Changsha 410073,China)

Abstract:With the rapid development of integrated circuit technology,the complexity of circuit design has been increasing,and transistor-level timing optimization has become an important means to improve circuit performance.To address the issue of excessive delay on critical paths in complex circuits composed of standard cells,a transistor-level timing optimization algorithm based on circuit topology has been proposed.This algorithm analyzes the topological characteristics of the circuit,accurately identifies the pull-up or pull-down paths through which current flows,and combines various optimization strategies to finely adjust the transistor sizes,thereby effectively reducing the delay of critical paths.Experiments have demonstrated the remarkable timing optimization effectiveness of this algorithm across various circuit scenarios.Specifically,under a comprehensive optimization strategy targeting all cells on the path,the pre-simulation test results of 200 test paths showed an average delay reduction of 20.7%;when the optimization focus was concentrated on delay-sensitive cells,these 200 paths still achieved an average delay reduction of 8.1% in pre-simulation tests.Furthermore,on a selected 10 paths,by optimizing only the delay-sensitive cells and completing layout drawing,post-simulation test results indicated an average delay reduction of 6.8% for these paths.These significant optimization achievements not only fully prove the effectiveness and practicality of the algorithm but also provide new insights for future locally fully-customized circuit design targeting delay-sensitive cells.

Keywords:transistor;circuit topology;circuit simulation;timing optimization

0 引 言

高性能中央处理器(CPU)追求降低延迟以提升主频,而寄存器间最长路径的过大延时则会限制主频的提升.当前,大多数 CPU 采用基于门级标准单元的半定制设计方法来构建,尽管标准单元库[1]和自动布局布线工具[2]极大地简化了电路设计流程,允许设计师根据需求从单元库中挑选合适的标准单元,但门级标准单元的尺寸变化并不连续,所选单元尺寸往往并非最优,这可能导致关键路径出现时序违例.在芯片签核(Signoff)阶段,静态时序分析依然会指出关键路径的时序

收稿日期:2025-02-20 收修改稿日期:2025-04-14 作者简介:**成泽祥**,男,2000 年生,硕士研究生,研究方向为超大规模集成电路时序分析及优化;**罗元盛**,男,1980 年生,博士,讲师,CCF 会员,研究方向为服务计算;**冯超超**(通信作者),男,1982 年生,博士,副研究员,CCF 会员,研究方向为高性能微处理器设计、片上网络和机器学习驱动的集成电路物理设计;**赵振宇**,男,1973 年生,博士,研究员,CCF 会员,研究方向为高性能微处理器物理设计和机器学习驱动的物理设计;**张曾慧**,女,1996 年生,硕士,工程师,研究方向为集成电路全定制设计;**成 龙**,男,2000 年生,工程师,研究方向为集成电路全定制设计.

违例,迫使电子工程师必须对关键路径中的标准单元版图设计[3]进行调整以降低延时.若能在晶体管级和版图级进行深入设计与优化,将有望实现主频的大幅提升.然而,在关键路径的优化实践中,传统方法在选择待优化单元、晶体管及其尺寸时,主要依赖于设计师的个人经验,未能充分挖掘电路拓扑结构中的有用信息.

模拟电路优化领域的研究为数字电路提供了重要方法论参考,在自动布局方面,Gao 等人[4]开发了一个交互式的模拟电路布局编辑系统,旨在缩小自动化布局与设计师手动布局之间的差距,从而提升模拟电路设计的效率和性能.在电路拓扑方面,Wang 等人[5]提出了图卷积神经网络-强化学习(GCN-RL)自动晶体管尺寸设计方法,实现了不同工艺节点和拓扑结构间的知识迁移.Li 等人[6]提出了基于电路注意力网络(CAN)的模拟电路晶体管尺寸优化方法,旨在自动化确定模拟电路设计中的晶体管尺寸,以提升电路性能.Uhlmann 等人[7]提出了使用深度强化学习(DRL)来解决模拟集成电路尺寸设计问题的方法.以上方法虽在技术路线上具有借鉴价值,但其面向连续参数的优化特性与数字电路离散化尺寸调整存在本质差异.

在传统 CMOS 晶体管优化领域中,在逻辑级优化方面,Shivam 等人[8]基于逻辑功效理论确定 PMOS 与 NMOS 最佳宽度比例,重点关注功率延迟积(PDP)最小化,但缺乏对 FinFET 器件物理特性的考量.Ahmed 等人[9]针对数字标准单元设计,提出一种 PVT 变化感知的晶体管尺寸优化方法,适用于低功耗、高性能设计.Kalluru 等人[10]提出一种基于萤火虫群优化(GSO)和邻域培育遗传算法(NCGA)的晶体管级优化方法,在工艺、电压、温度与老化(PVTA)综合变化下优化功耗与时序性能.在电路设计方面,Kunwar 等人[11]结合逻辑功效理论(LE)与启发式算法(ISA/GSA)改变晶体管宽度,优化电路的功率延迟面积积(PDAP).在漏电功耗方面,Gupta 等人[12]则提出了一种基于优化算法的鲁棒晶体管尺寸设计方法,利用模拟退火算法和人工蜂群算法优化晶体管尺寸,以最小化漏电功耗.此外,在抗辐照电路设计领域,王海滨等人[13]提出了一种基于双 TPDICE 环形结构的抗三节点翻转触发器,其通过两级 C 单元错误拦截装置和时钟堆栈架构,降低了特定电路结构的延时.针对 FinFET 器件特性的研究也取得显著进展,在物理实现层,Xu 等人[14]提出布局参数协同优化方法,以多晶硅间距等物理参数为优化对象,通过设计技术协同优化(DTCO)实现功耗、性能、面积(PPA)权衡,满足标准单元库的多样化设计需求.

当前研究在通过深入分析电路拓扑结构来调整门级标准单元版图尺寸方面尚显不足.综上所述,尽管现有的关键路径优化方法已取得显著成效,但它们对于国产高性能处理器的具体优化效果仍有待验证.鉴于不同应用场景下的需求差异,本文亟需从实际工程项目出发,依据调整标准单元尺寸后的仿真延时数据,从修改单元网表和版图设计的角度,深入探究影响延时的关键因素.因此,深入探究电路拓扑结构、研究标准单元尺寸的优化策略,以降低关键路径延时,不仅具有理论价值,更具备迫切的现实意义.

本研究受传统电路中通过减小电阻以增强电流流通效率的启发,提出了一种基于电路拓扑结构的晶体管级时序优化算法.该算法巧妙融合不同单元类型与电路拓扑结构的特性,分析电流流经的上拉或下拉路径,并据此构建了一个输入输出与电流流经晶体管之间的对照表.基于这一对照表,算法能够精准地调整单元网表中的晶体管宽度,以实现性能优化.为了验证算法的有效性,本文对优化前后的路径进行了前仿真,通过对比分析,精准识别出延时减少最为显著的 5 个延时敏感单元.随后,针对这些延时敏感单元,按照优化后的晶体管尺寸重新绘制了版图,并进行了后仿真验证.这一方法在确保改动单元数量最少的同时,最大限度地优化了关键路径的延时,为解决电路设计中的关键路径时序违例问题提供了指导,包括明确指出了需要修改的单元位置、晶体管位置以及晶体管尺寸,为电路设计优化开辟了新路径.

1　相关工作

随着晶体管尺寸的持续缩减,金属氧化物半导体场效应晶体管(MOSFET)因短沟道效应而面临电流控制失效的挑战[15],在此背景下,鳍式场效应晶体管(FinFET)凭借其出色的栅极控制能力[16],成为替代传统平面 MOSFET 的优选方案,有效遏制了电流的泄露问题.值得注意的是,FinFET 逻辑门的有效电容与链级数的关系并非直观的正比关系,而是与输入输出节点的转换时间紧密相关.因此,传统的平面逻辑电路中用于确定晶体管尺寸的方法,在 FinFET 逻辑电路设计中已不再适用[17].本研究聚焦于将电路拓扑结构作为优化算法的核心驱动力,致力于寻找关键路径中晶体管的最佳尺寸配置,并通过前仿真与部分后仿真的方式,全面评估优化前后的关键路径性能,以期实现设计的最优化.

复杂路径延时的评估可通过静态时序分析[18]和动态电路仿真[19,20]两种方法实现.静态时序分析以其高效性著称,但所得结果往往偏于保守;相比之下,动态电路仿真(例如使用 SPICE)虽然耗时较长,却能提供更精确的延时评估,尤其适用于关键路径延时的细致考量.为了精确测量电路延时,需细致分析仿真波形的上升沿与下降沿特征.本研究提出了一种自动化测量方法,该方法通过巧妙嵌入测量语句并直接导出延时测量结果,在提升效率的同时确保了测量精度.在晶体管级时序优化领域,已涌现出多种高效策略,它们大致可以分为两大类:基于逻辑重组的优化和基于物理设计的优化.基于逻辑重组的优化,其核心在于通过调整电路的逻辑结构来达到减少延时的目的.这种优化方式往往需要对电路的逻辑功能有深入的理解,并能精准地识别出那些对延时影响最为关键的逻辑环节,进而实施针对性的调整.基于物理设计的优化,则更多地聚焦于如何通过优化电路的布局与布线,来进一步优化信号的传播路径.这包括但不限于对晶体管、互连线以及电源、地网络的布局进行精细调整,以减少信号传输过程中的损耗和延时.特别地,在处理关键路径上的标准单元版图时,采用定制单元版图的方式往往能够取得更为显著的效果.通过针对特定应用场景和性能需求,对单元版图进行量身定制,可以在物理设计层面实现更为精细和高效的优化,从而进一步提升电路的整体性能和稳定性.

集成电路设计方法主要分为全定制设计、半定制设计和可编程逻辑器件设计.全定制设计[21]是一种高度个性化的设

计方法,它依赖于设计者的深厚经验和精湛技艺,通过人工对电路结构和参数进行细致入微的优化,力求达到最佳性能和最小芯片尺寸的双重目标. 这种方法通常用于对性能要求极高或对芯片尺寸有严格限制的场合. 半定制设计[22]则在设计过程中更加注重效率和灵活性. 它允许设计者在设计阶段从标准单元库中选择合适的单元和模块进行组合,从而大大缩短了设计周期,降低了设计成本. 同时,由于采用了标准化的单元和模块,半定制设计在制造过程中也更容易实现规模效应,进一步降低了生产成本. 可编程逻辑器件设计[23,24]则是一种更为灵活多变的设计方法. 它利用可编程逻辑门阵列(FPGA)等可编程器件,通过配置逻辑门的连接方式来实现不同的功能. 这种方法使得设计者能够在不改变硬件电路的情况下,通过软件编程来快速实现电路功能的调整和优化,从而大大提高了设计的灵活性和可重用性. 在高性能计算领域,由于需要处理的数据量巨大且计算任务复杂,因此芯片的性能往往成为决定系统整体性能的关键因素. 为了满足这一需求,高性能计算领域的芯片多采用全定制或半定制设计流程. 其中,全定制设计能够针对特定应用场景进行深度优化,从而实现极致的性能表现;而半定制设计则能够在保证一定性能水平的同时,提供更高的设计效率和更低的成本. 值得注意的是,与全定制设计相比,对延时敏感单元进行局部全定制优化往往能够取得更为显著的性能提升效果. 这是因为局部全定制优化能够针对关键路径中的延时瓶颈进行精准打击,通过优化单元、晶体管及其尺寸等参数来有效缩短信号传输时间[25]. 在选择要优化的单元、晶体管及其尺寸时,设计者需要综合考虑电路拓扑结构、信号传输路径以及性能需求等多个因素,以确保优化方案的有效性和可行性.

电路拓扑结构清晰地描绘了电路中各个元件之间的连接关系[26]. 本文以此为出发点,深入分析了输入信号的上升沿与下降沿如何触发电路中电流上拉或下拉路径的动态变化,而且进一步研究了输入信号变化对门级单元内部上拉或下拉路径的具体影响,并在此基础上,设计了一种晶体管级延时优化算法. 该算法能够依据实际需求,动态地调整晶体管鳍片(fin)的数量,从而实现对晶体管级延时性能的优化. 为了验证这一时序优化方法的有效性,本文特意选取了实际工程项目中的两个模块,涵盖了共计 200 条关键路径进行详细的分析与测试. 测试结果表明,当对路径上所有单元进行全面优化后,这 200 条路径的前仿测试性能平均提升了 20.7%. 而仅针对延时敏感单元进行优化时,相关路径的前仿测试性能也平均提升了 8.1%. 更进一步地,在 10 条精选路径上仅对延时敏感单元进行优化并绘制了相应的版图后进行后仿测试,结果显示性能平均提升了 6.8%. 这些显著的性能提升充分证明了本文所提时序优化方法的实用性和有效性.

2 基于电路拓扑结构的晶体管级时序优化算法

2.1 FinFET 晶体管级时序优化原理

MOSFET 晶体管的时序与导通电阻和栅极电容相关,而导通电阻和栅极电容在沟道长度 L 固定的情况下,则取决于沟道宽度 W. 传统的平面 MOSFET 晶体管的导通电阻 R 计算公式[27]和栅极电容 C_G 计算公式[27]分别见式(1)和式(2):

$$R = R_{eq}\frac{L}{W} \tag{1}$$

$$C_G = C_{ox}WL \tag{2}$$

其中 R_{eq} 是等效电阻,C_{ox} 是栅氧化层电容,L 是沟道长度,W 是沟道宽度.

平面 MOSFET 晶体管的 W 可以通过调整版图来直接改变,对于单元中电流经过的晶体管增加 W,可以减小 R,降低延时,对于电流不经过的晶体管减小 W,可以减小 C_G,抵消因增加 W 导致的电容增加,但是 FinFET 晶体管因其独特的制造工艺,其中栅电极的长度(l)、fin 的高度(h)以及 fin 的宽度(w)在制造过程中均被固定,唯一可调整的参数是鳍片(fin)的数量. 通过增加门级单元内部电流流经的上拉或下拉路径上的晶体管鳍片数量,实质上等同于拓宽了 MOS 晶体管的沟道宽度 W,这一改变能够显著降低导通电阻 R,进而增大电流,有效减少电路延时[28]. 相反地,对于不参与电流传导的上拉或下拉路径上的晶体管,减少电流不经过上拉或下拉路径上的晶体管鳍片数量,则相当于缩减了 MOS 晶体管的宽度 W,这一举措有助于降低栅寄生电容 C_G. 这样做的好处在于,它能够部分或全部抵消因增加电流路径晶体管鳍片数量所带来的电容增加,从而减轻对前后级单元延时的不利影响[29].

在版图实现中,调整晶体管鳍片(fin)数量的常用方法是通过改变栅极的数量来间接实现. 每个单元内的晶体管的栅级都预设了一定数量的鳍片,文中的栅极结构包含 5 根鳍片. 增加栅极的数量,实质上就是按照每个栅极所含鳍片的数量来增加总的鳍片数,这样做能够拓宽 MOS 晶体管的沟道宽度,进而减小导通电阻 R,达到降低延时的效果. 相反地,减少栅极数量则会相应地缩减沟道宽度,并有助于降低栅极寄生电容 C_G. 通过直接调整栅极数量来按比例改变鳍片数量,在版图绘制过程中更为直观且易于操作. 每增加 1 个栅极,就相当于增加了 5 根鳍片,从而在整体上加宽了 MOS 晶体管沟道宽度. 这种方法不仅简化了设计流程,还确保了性能调整的准确性和可控性.

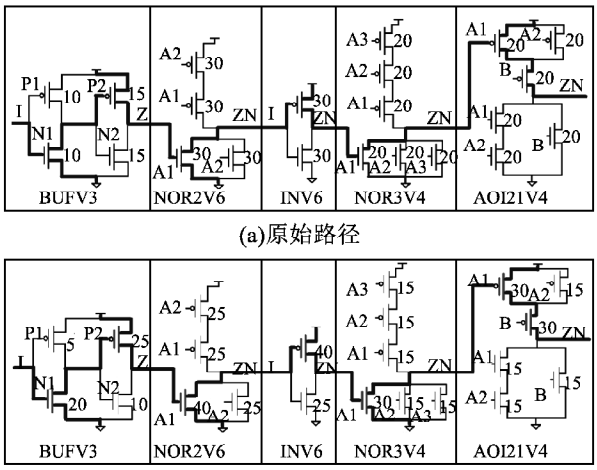
2.2 确定电流经过的上拉或下拉路径

静态时序分析能够识别出延时最长的关键路径,该路径依据电路的具体拓扑结构以及逻辑门输入信号的上升沿或下降沿来确定电流的上拉或下拉走向. 以图 1(a)所示路径为例,图中数字代表了各晶体管中的 fin 数量,其中加粗连线明确指出了电流流经的关键路径(即延时最长的路径),而未加粗的连线则代表电流不经过的非关键路径. 进一步地,图 1(b)展示了路径优化后的结果:关键路径上的晶体管通过增加 2 根栅极,达到了增加 10 根 fin 的效果,以增强其性能;与此同时,非关键路径上的晶体管则减少了 1 根栅极,即减少 5 根 fin,以实现资源的有效分配.

2.3 时序优化算法框架

为了优化电路中关键路径上的晶体管尺寸,本研究首先分析了晶体管级电路的单元拓扑结构,并设计了一个对照表,该表详细记录了逻辑门的输入输出信号与上拉或下拉路径上晶体管之间的对应关系. 以图 2 中展示的 AO222 电路结构为例,具体的对照信息参见表 1,其中涵盖了信号的上升(r)和下降(f)两种情况. 该电路存在 12 种不同的输入输出信号组

合情形. 举例来说,当输入信号 A2 处于上升沿(r)且输出信号 Z 也处于上升沿(r)时,电流会流经晶体管 N3、N6 和 P7,



(a)原始路径

(b)优化路径

图 1　查找电流经过的上拉或下拉路径

Fig. 1　Find the pull-up or pull-down path through which the current flows

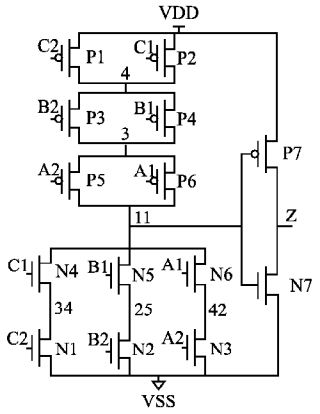


图 2　AO222 电路图

Fig. 2　AO222 circuit diagram

而不会流经晶体管 P1 ~ P6 以及 N1、N2、N4、N5 和 N7. 基于这一对照表,时序优化算法能够对电流流经的上拉或下拉路径上的晶体管增加栅极数量,同时减少电流不流经的路径上晶体管的栅极数量,从而有效实现时序优化. 值得注意的是,对于标准单元库中的每一个单元,仅需建立一次对照表即可实现重复使用,这相当于在单元网表与其电路拓扑之间建立了一个便捷的映射关系. 未来,还可以进一步引入针对网表自动识别电路拓扑的先进算法. 此外,算法中增加或减少的栅极数量是可以灵活控制的,这使得算法的适用范围更加广泛. 具体而言,该算法主要适用于以下场景:关键路径已经明确确定,单元类型、单元级数以及电路逻辑均不可更改,仅能对关键路径上的单元版图进行必要调整的情况.

时序优化算法的流程如图 3 所示. 首先,算法会获取待优化时序路径上的全部 N 个单元,并针对电流流经的上拉或下拉路径上的晶体管预设增加栅极的数量,同时对于电流不流经的路径上的晶体管预设减少栅极的数量. 随后,算法会查询并获取每个单元内部各个晶体管的相关参数信息. 依据输入输出信号变化情况与待优化晶体管的对照表,算法会对每个单元内的晶体管栅极数量进行调整,从而生成优化后的时序路径. 紧接着,对优化后的时序路径进行仿真,以获取优化后

表 1　AO222 电路工作状态对照表

Table 1　AO222 circuit working status comparison table

输入	输出			上拉或下拉路径上的晶体管	
				电流经过	电流不经过
A2	r	Z	r	N3,N6,P7	P1,P2,P3,P4,P5,P6,N1,N2,N4,N5,N7
A2	f	Z	f	P1,P2,P3,P4,P5,N7	N1,N4,N2,N5,N3,N6,P6,P7
A1	r	Z	r	N3,N6,P7	P1,P2,P3,P4,P5,P6,N1,N2,N4,N5,N7
A1	f	Z	f	P1,P2,P3,P4,P6,N7	N1,N4,N2,N5,N3,N6,P5,P7
B2	r	Z	r	N2,N5,P7	P1,P2,P3,P4,P5,P6,N1,N3,N4,N6,N7
B2	f	Z	f	P1,P2,P3,P5,P6,N7	P4,P7,N1,N2,N3,N4,N5,N6
B1	r	Z	r	N2,N5,P7	P1,P2,P3,P4,P5,P6,N1,N3,N4,N6,N7
B1	f	Z	f	P1,P2,P4,P5,P6,N7	P3,P7,N1,N2,N3,N4,N5,N6
C2	r	Z	r	N1,N4,P7	P1,P2,P3,P4,P5,P6,N2,N3,N5,N6,N7
C2	f	Z	f	P1,P3,P4,P5,P6,N7	P2,P7,N1,N2,N3,N4,N5,N6
C1	r	Z	r	N1,N4,P7	P1,P2,P3,P4,P5,P6,N2,N3,N5,N6,N7
C1	f	Z	f	P2,P3,P4,P5,P6,N7	P1,P7,N1,N2,N3,N4,N5,N6

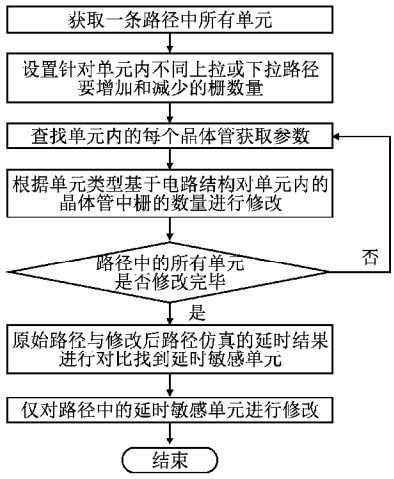


图 3　时序优化算法流程图

Fig. 3　Flowchart of timing optimization algorithm

的路径延时,并从中筛选出延时变化最为显著的 n 个单元(其中 $n \leqslant N$),这些单元被视为延时敏感单元. 最后,针对这些筛选出的延时敏感单元,按照时序优化的具体修改要求重新绘制版图. 版图绘制完成后再进行一次后仿真,以确保优化效果得以实现. 通过这一系列流程,算法能够基于电路拓扑结构实现晶体管级的时序优化,从而快速且高效地缩减关键路径的延时.

2.4 获取标准单元晶体管尺寸算法

获取标准单元晶体管尺寸的算法如下：

算法 1. 获取标准单元晶体管尺寸算法

输入：源 *source*，栅 *gate*，漏 *drain*，晶体管类型 *mos_type*

输出：单元内每个晶体管中含有栅的数量 *gate_num*，一根栅含有 fin 的数量 *nfin*，每个晶体管含有 fin 的总数 *all_fin_num*

```
1. while ( loop ! =0 ) {
2.   cell_line = 单元网表中第 cell_line_num 行;
3.   cell_line 中截取 word1,word2,word3,word4;
4.   if (word1 == mos_type && word2 == source && word3 == gate
&& word4 == drain ) {
5.     nfin = cell_line 中鳍片 fin 的数量;gate_num + + ;}
6.     else cell_line_num = cell_line_num + 1;
7.     if (cell_line == . ENDS) {
8.     loop = 0;all_fin_num = gate_num * nfin;}}
```

本算法旨在处理单元的标准单元网表数据，以解析出单元内各晶体管的关键参数. 其输入信息包括晶体管的源(*source*)、栅(*gate*)、漏(*drain*)以及晶体管类型(*mos_type*，即 N 型或 P 型). 算法的输出结果包括每个晶体管含栅极的数量 *gate_num*、每根栅极上含有的鳍片数量 *nfin*，以及每个晶体管鳍片的总数 *all_fin_num*. 此外，算法引入了一个变量 *cell_line_num*，用于追踪并读取单元网表中的行号，其初始值被设定为 2. 另一变量 *loop* 作为循环的控制器，当 *loop* 值为 1 时，循环继续执行；当 *loop* 值为 0 时，循环终止.

算法具体流程如下：从当前单元的标准单元网表中，依据行号 *cell_line_num* 逐一读取行内容 *cell_line*. 检查读取的行内容 *cell_line* 是否同时包含源、栅、漏以及晶体管类型这 4 个关键要素，并且与正在查找的晶体管类型相匹配. 若条件满足，则进一步从该行中提取出每根栅极所含鳍片的数量 *nfin*，并继续读取下一行的内容. 记录满足上述条件的行数作为当前单元内晶体管含栅极的数量 *gate_num*. 根据提取的每根栅极所含鳍片的数量 *nfin*，以及计算得到的晶体管所含栅极的数量 *gate_num*，二者相乘即可得出每个晶体管中鳍片的总数 *all_fin_num*.

2.5 晶体管加减栅判断算法

晶体管加减栅判断算法如下：

算法 2. 晶体管加减栅判断算法

输入：加或减栅 *judge*，加或减的栅数 *gate_change*，栅含有 fin 的数量 *nfin*，栅的总数 *gate_num*，源 *source*，栅 *gate*，漏 *drain*，晶体管类型 *mos_type*

输出：晶体管加减栅后的单元网表文件

```
1. if (judge == 1 ) {
2. need_add_num = gate_change * nfin;add_fin = need_add_num + nfin;
3.   while ( need_add_num ! == 0 ) {
4.   cell_line = 单元网表中第 cell_line_num 行;
5.   cell_line 中截取 word1,word2,word3,word4;
6.   if ( word1 == mos_type && word2 == source && word3 == gate &&
word4 == drain) {
7.   if ( add_fin < = max_fin_num) {
8.   cell_line 中鳍片 fin 的数量 = add_fin;need_add_num = 0;}
9.   else { cell_line 中鳍片 fin 的数量 = max_fin_num;
10. need_add_num = (need_add_num - max_fin_num) + nfin;
11. add_fin = need_add_num + nfin;cell_line_num = cell_line_num +
1;} }
12. else cell_line_num = cell_line_num + 1;
13. if ( cell_line ==. ENDS ) {printf ( " no place to add gate!" );
break;} } }
14. if ( judge == 0 ) {
15. if ( gate_change > = gate_num ) { printf ( " no place to minus
gate!" );break;}
16. else {
17. while ( gate_change ! == 0 ) {
18. cell_line = 单元网表中第 cell_line_num 行;
19. cell_line 中截取 word1,word2,word3,word4;
20. if (word1 == mos_type && word2 == source && word3 == gate &&
word4 == drain) {
21. 删除这一行;
22. gate_change = gate_change - 1;}
23. else cell_line_num = cell_line_num + 1;
24. if ( cell_line == . ENDS ) {printf ( " no place to minus gate!" );
break;} } } }
```

在该算法中，*cell_line_num* 是一个用于追踪单元网表文件中当前行号的变量，*need_add_num* 代表需要向晶体管中额外添加的鳍片总数，而 *add_fin* 则用于调整每根栅极上的鳍片数量. 算法通过 *judge* 变量来决定是进行栅极的增加还是减少操作：当 *judge* 为 1 时，表示需要增加栅极；当 *judge* 为 0 时，则表示需要减少栅极. 涉及的关键变量还包括 *gate_change*(增或减的栅极数量)、*nfin*(每根栅极所含的鳍片数量)、*gate_num*(晶体管中栅极的总数)、*source*(源极)、*gate*(栅极)、*drain*(漏极)、*mos_type*(晶体管类型，即 N 或 P 型). 算法的输出是一个经过栅极调整后的晶体管单元网表文件.

对于加栅操作，算法首先确定当前单元中，在电流的上拉或下拉路径上，各晶体管所需增加的栅极数量 *gate_change*. 随后，根据 *gate_change* 和每根栅极原有的鳍片数量 *nfin*，计算出总共需要增加的鳍片数量 *need_add_num*. 若 *need_add_num* 不为 0，算法则逐行读取单元网表中的 *cell_line*，寻找与源极、栅极、漏极以及晶体管类型相匹配的行. 对于每一匹配行，算法会检查在增加 *need_add_num* 个鳍片后，每根栅极的鳍片数量是否仍不超过预设的最大值 *max_fin_num*. 若增加后的鳍片数量不超过 *max_fin_num*，则更新该行的 *nfin* 值，并将 *need_add_num* 置为 0，表示增加的鳍片数量已满足要求. 若增加后的鳍片数量超过 *max_fin_num*，则将该行的 *nfin* 设置为 *max_fin_num*，并重新计算 *need_add_num*(即 *need_add_num* 减去已添加到当前行的最大鳍片数后，再加上原 *nfin* 值)，随后继续读取下一行进行处理.

对于减栅操作，算法会确定当前单元中，在电流不经过的上拉或下拉路径上，各晶体管所需减少的栅极数量 *gate_change*. 若 *gate_change* 大于或等于当前晶体管栅极的总数 *gate_num*，则表明无需进一步优化. 若 *gate_change* 不为 0 且小于 *gate_num*，算法则逐行读取单元网表中的 *cell_line*，寻找与源极、栅极、漏极以及晶体管类型相匹配的行，并将其删除. 每删除一行，*gate_change* 就相应减少，直至 *gate_change* 为 0，表示减栅操作已完成. 通过上述流程，算法能够精准地实现晶体管栅极的增加与减少操作，从而满足特定的电路设计与优

化需求.

3　实验结果及分析

本文选取实际工程项目中两个模块所涉及的 200 条关键路径作为研究对象. 为了优化这些路径的延迟,采用了文中所提出的时序优化算法,该算法通过增减路径上晶体管的栅极数量来达到优化目的. 实验运行在一台高性能的 Linux 服务器上,该服务器配置了 160 个 Intcl Xcon 2.5GHz 的 CPU 核心以及 1TB 的内存. 为了全面评估优化算法的效果,采用了业界主流的商用 SPICE 仿真软件,对优化前后的关键路径分别进行了前仿真和后仿真. 这一系列的仿真步骤旨在确保能够准确衡量并验证算法所带来的性能提升.

3.1　增加不同数量栅对路径延迟的影响

为了深入探究栅极数量增加对路径延迟的具体影响,本研究精心挑选了 10 条路径进行实验,针对这些路径上的晶体管,根据算法的判断,分别在上拉或下拉电流路径上增加了 1 ~5 根栅极,并进行了前仿真以评估效果. 实验的结果详细记录在表 2 中. 通过观察优化百分比随栅极数量增加的变化趋势,可以清晰地看到,栅极数量的增加与性能优化效果呈现出正相关关系,即栅极增加得越多,优化效果越显著. 然而,从版图设计的实际角度出发,过多的栅极会导致芯片面积的浪费,

表 2　关键路径上晶体管增加不同数量栅后的延时(ps)

Table 2　Delay of transistors on critical paths with different number of gates added(ps)

路径名称	原始	加 1	加 2	加 3	加 4	加 5
Path1	150.5	136.3	128.7	119.4	114.2	110.0
Path2	157.9	144.2	135.6	127.6	122.5	118.6
Path3	151.4	137.6	130.1	121.2	116.3	112.2
Path4	153.5	140.0	131.9	123.8	118.7	114.7
Path5	153.8	139.7	131.2	122.9	117.6	113.5
Path6	153.0	139.5	131.4	123.3	118.1	113.3
Path7	153.6	139.5	130.9	122.7	117.4	113.2
Path8	151.8	139.7	128.8	122.9	115.1	110.8
Path9	151.8	137.5	129.1	120.8	115.6	111.5
Path10	153.5	139.9	131.9	123.8	118.7	114.7
优化百分比		8.9%	14.4%	19.7%	23.3%	26.0%

而仅仅增加 1 根栅极的优化效果又相对有限. 因此,在综合权衡晶体管面积占用与版图设计的易实现性之后,做出了一个折中的决策:为每条路径上的晶体管增加 2 根栅极. 这样的调整旨在实现路径延迟的优化,同时尽量保持版图设计的简洁与高效.

3.2　修改不同单元对路径延迟的影响

本研究深入探索了 3 种不同的优化策略对同一条路径延迟改善的效果,这 3 种策略分别是:全面优化路径上的所有单元、单独优化路径中的单个单元,以及有针对性地对 5 个延时高度敏感的单元进行优化. 如图 4 所示,在单独优化单个单元的尝试中,发现时序优化效果最为突出的延时敏感单元分别是编号为 3、5、8、11 和 18 的单元,这一发现与全面优化所有单元后识别出的延时敏感单元完全吻合. 鉴于此发现,为了提升优化效率,在后续的优化流程中,摒弃了单独优化单个单元的方法,转而采取了一种更为高效的策略:首先对整个路径上的所有单元进行一次性全面优化,随后在这一基础上精确识别出延时敏感单元. 这种策略旨在通过减少优化迭代的次数,来更加高效地实现路径延迟的优化. 仅优化 5 个延时敏感单元即可显著降低路径延迟,表明针对特定敏感单元进行精准优化,能够有效实现关键路径的时序优化目标.

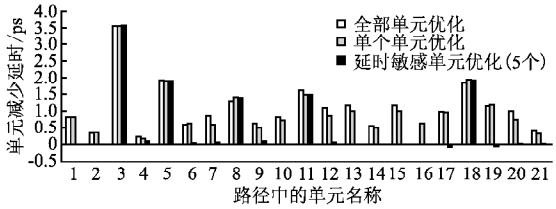
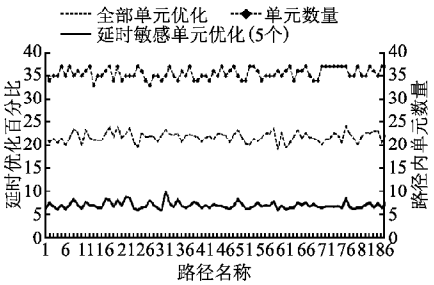


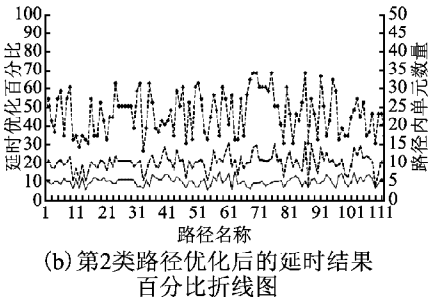
图 4　不同方式优化的延时减少柱状图

Fig. 4　Histogram of delay results of a path optimized in different ways

图 5 直观地展示了针对 200 条路径实施优化措施后的成效. 这些路径被划分为两大类:第 1 类路径的特点是,它们属于同一种类型,但单元的组合方式各异. 在这类路径中,每条路径大致包含 35 个单元. 当对所有单元进行全面优化后,路径的延时改善率显著,达到了 20% ~25% 的范围. 而如果仅选择优化其中 5 个延时最为敏感的单元,那么延时改善的效果则相对有限,大约在 5% ~10% 之间. 第 2 类路径则包含了



(a) 第1类路径优化后的延时结果百分比折线图



(b) 第2类路径优化后的延时结果百分比折线图

图 5　200 条路径优化后的延时结果百分比折线图

Fig. 5 Percent delay result line graph for 200 optimized paths

不同类型的路径,这些路径中的单元数量介于 15 ~35 个之间. 对于这类路径,全面优化所有单元所带来的延时改善率会根据单元数量的不同而有所变化,改善率范围大致在 10% ~30% 之间. 同样地,如果仅针对延时敏感的单元进行优化,其效果则相对稳定,约为 10% 的延时改善.

3.3　修改单元版图对路径延迟的影响

从 200 条路径中精心挑选了 10 条进行延迟敏感单元的针对性优化,并在完成版图设计后进行了后仿真测试,表 3 详

细列出了这 10 条路径的前仿真和后仿真结果. 测试结果表明,尽管在引入寄生参数的影响下,后仿真的优化效果相较于前仿真略有降低,但它更贴近于实际的应用场景. 这一结果有力地验证了仅通过优化延时敏感单元就能实现显著效果的观点. 统计显示,这些路径平均包含 27 个单元. 当对所有单元进行全面优化时,前仿真的平均优化百分比高达 23.9%;而当仅针对延时敏感单元进行优化时,前仿真的平均优化百分比为 11.3%,在后仿真中这一数值降至 6.8%. 尽管后仿真的优化效果有所减少,但它依然证明了延时敏感单元优化策略的有效性和实用性.

表 3 10 条路径前仿真和后仿真实验结果

Table 3 Results of pre-simulation and post-simulation experiments for 10 paths

路径名称	单元数量	原始网表		全部单元优化		延时敏感单元优化(5 个)			
		前仿延时(ps)	后仿延时(ps)	前仿延时(ps)	前仿优化百分比	前仿延时(ps)	前仿优化百分比	后仿延时(ps)	后仿优化百分比
1	35	155.7	393.6	119.3	23.3%	140.0	10.0%	372.0	5.4%
2	35	153.5	392.5	117.2	23.6%	140.3	8.5%	374.4	4.6%
3	35	155.8	391.1	120.9	22.4%	142.1	8.7%	373.0	4.6%
4	37	157.4	397.1	119.3	24.2%	144.2	8.3%	375.0	5.5%
5	35	151.8	394.1	116.1	23.5%	139.2	8.3%	372.0	5.6%
6	18	134.7	270.1	105.1	21.9%	115.3	14.4%	251.7	6.8%
7	21	86.5	231.2	60.4	30.0%	74.6	13.6%	213.9	7.4%
8	17	134.2	262.2	105.9	21.0%	115.1	14.2%	234.6	10.5%
9	20	111.6	237.3	87.2	21.8%	96.3	13.7%	215.4	9.2%
10	24	117.5	284.1	84.9	27.6%	101.0	14.0%	259.2	8.7%
平均	27				23.9%		11.3%		6.8%

本研究还进一步对延时敏感单元的使用频次进行了详尽的统计分析. 图 6 直观展示了第 2 类路径中所有延时敏感单元的类型及其出现频次. 经过仔细识别,共发现了 44 种不同的单元类型,其中,单元类型 1 使用频次高达 32 次,单元类型 30 具备 6 种不同的工作状态. 在柱状图中,同一柱体内颜色相同的部分代表了工作状态相同的单元. 这一发现意味着,对于具有相同类型和工作状态的单元,可以采用统一的优化策略. 通过实施一次性优化并绘制相应的版图,能够实现这些单元的复用,这一举措显著提升了对不同路径的优化效率,并有效降低存在时序违例的关键路径延迟. 这一发现不仅为优化工作提供了宝贵的指导,也为后续的研究和实践奠定了坚实的基础.

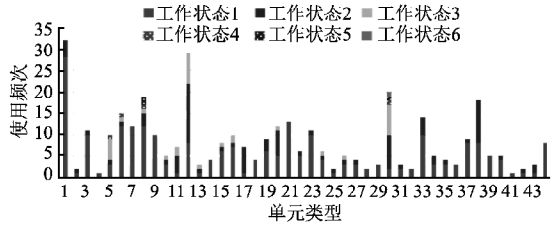


图 6 第 2 类路径的所有延时敏感单元的类型及使用频次堆积柱状图

Fig. 6 Cell stacked histogram of all delay sensitive cells and the usage frequency in the second path

3.4 算法复杂度与横向对比分析

本算法的优化代价主要体现在版图修改的复杂性和计算资源消耗. 在时间复杂度上,设路径包含 K 个单元,每个单元网表平均 M 行. 算法需对每个单元调用算法 1 和算法 2,算法 1 的时间复杂度是 $O(M)$,算法 2 的时间复杂度是 $O(M)$,整体流程的时间复杂度为 $O(K \cdot M)$. 其中网表行数 M 为常数,则简化为 $O(K)$,即线性于路径单元数. 在空间复杂度上,对照表预处理后固定,运行时仅需临时变量,空间与输入规模无关,所以为 $O(1)$. 建立对照表属于预处理,单元类型与拓扑的映射关系为静态数据,仅需构建一次,不纳入算法运行时复杂度. 仿真时,SPICE 仿真为外部工具,其耗时独立于算法复杂度分析. 综上,该算法在时间和空间上均具备高效性,适用于大规模集成电路关键路径的晶体管级时序优化.

为体现方法学创新性,将本文算法与主流方法进行了对比. 如表 4 所示,相较传统方法,本文的优化对象是晶体管鳍片数量,核心是在晶体管中通过改变栅的数量来改变 fin 的数量,从而影响导通电阻及寄生电容参数,技术特征是 FinFET 物理拓扑分析,适用场景是后端局部路径修复. 当前阶段,本文通过 200 条工业级设计路径的严格验证(含前仿、后仿对比),已充分证明方法在真实设计场景中的有效性.

表 4 算法对比

Table 4 Algorithm comparison

来源	方法	优化层级	工艺支持	仿真验证	面积代价	适用场景
文献[8]	逻辑功效优化	门级逻辑优化	平面 MOSFET	SPICE 前仿	全局调整	基于逻辑功效的单元功率延迟积(PDP)优化
文献[9]	模因算法	晶体管级	平面 MOSFET	SPICE 前仿	全局调整	低功耗设计、鲁棒性提升、面积缩减
文献[10]	GSO,NCGA 算法	晶体管级	平面 MOSFET	SPICE 前仿,后仿	全局调整	考虑 PVTA 鲁棒性的低功耗、高性能应用
文献[11]	LE-ISA,GSA 算法	门级,晶体管级	平面 MOSFET	SPICE 前仿	全局调整	优化电路功率延迟面积积(PDAP)
文献[14]	DTCO 方法	版图级,工艺级	FinFET	TCAD 仿真	全局调整	标准单元库的多样化设计需求
本文	电路拓扑驱动算法	晶体管级物理优化	FinFET	SPICE 前仿,后仿	局部修改	高性能芯片关键路径时序修复,局部全定制

4　结束语

本文创新性地提出了一种基于电路拓扑结构的时序优化策略,该策略巧妙融合了电路特性与优化算法,通过深入分析电路拓扑结构以及电流在上拉或下拉路径中的流动情况,来精确确定晶体管的尺寸,从而实现了电路性能的显著提升. 实验数据有力地证明了这一方法的有效性:在针对 200 条关键路径的优化过程中,仅通过调整 5 个延时敏感单元的网表,SPICE 前仿真的延时就平均降低了 8.1%;进一步地,当这些调整反映到版图上后,后仿真的延时也实现了 6.8% 的平均优化.

这一方法在局部全定制电路的时序优化及晶体管尺寸确定领域展现出了广阔的应用潜力. 尤为值得一提的是,该方法还能够与现有的机器学习预测模型进行无缝对接,从而进一步缩短获取仿真延时数据的时间,提高优化效率. 研究过程中还发现,若能在局部全定制的基础上,对路径中的更多单元进行细致的调整,即针对关键路径对更多的标准单元版图进行定制化处理,电路的延时有望降至最低水平. 展望未来,计划进一步探索该方法与其他先进优化技术的融合策略,并致力于将其应用于更为复杂、多样化的电路场景中,以期不断拓展其应用范围,提升电路设计的整体效能.

References:

[1] Suroshi V, Karthik B K, Kannur V, et al. Optimization of subthreshold standard cells for energy efficient designs [C]//Proceedings of the 38th International Conference on VLSI Design and 23rd International Conference on Embedded Systems (VLSID), 2025: 475-480.

[2] Gusmao A P L D, Horta N C G, Lourenco N C C, et al. Scalable and order invariant analog integrated circuit placement with attention-based graph-to-sequence deep models[J]. Expert Systems with Application, 2022, 207: 1-20, doi: 10.1016/j.eswa.2022.117954.

[3] SUN L L. Study on layout design skills of integrated circuit standard cells[J]. Applications of IC, 2022, 39(1): 13-15.

[4] Gao X H, Zhang H Y, Liu M J, et al. Interactive analog layout editing with instant placement and routing legalization [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023, 42(3): 698-711.

[5] Wang II R, Wang K, Yang J C, et al. GCN-RL circuit designer: transferable transistor sizing with graph neural networks and reinforcement learning [C]//Proceedings of the 57th ACM/IEEE Design Automation Conference, 2020: 1-6.

[6] Li Y G, Lin Y S, Madhusudan M, et al. A circuit attention network-based actor-critic learning approach to robust analog transistor sizing[C]//Proceedings of the ACM/IEEE 3th Workshop on Machine Learning for CAD, 2021: 1-6.

[7] Uhlmann Y, Essich M, Bramlage L, et al. Deep reinforcement learning for analog circuit sizing with an electrical design space and sparse rewards [C]//Proceedings of the ACM/IEEE 4th Workshop on Machine Learning for CAD, 2022: 21-26.

[8] Singh S, Ojha P K, Asati A R. Analysis of logical effort-based optimization in the deep submicron technologies[J]. Emerging Technology Trends in Electronics, Communication and Networking, Lecture Notes in Electrical Engineering, 2023, 952: 23-37, doi: 10.1007/978-981-19-6737-5_3.

[9] Ahmed M S, Abbas Z. A memetic algorithm based PVT variation-aware robust transistor sizing scheme for power-delay optimal digital standard cell design [C]//Proceedings of the IEEE 37th International Conference on Computer Design, 2019: 358-352.

[10] Kalluru H S, Saha P, Zahra A, et al. Algorithm driven power-timing optimization methodology for CMOS digital circuits considering PVTA variations [C]//Proceedings of the IEEE International Symposium on Circuits and Systems, 2021: 1-5.

[11] Singh K, Jain A, Mittal A, et al. Optimum transistor sizing of CMOS logic circuits using logical effort theory and evolutionary algorithms[J]. Intcgration thc VLSI Journal, 2018, 60: 25-38, doi: 10.1016/j.vlsi.2017.08.003.

[12] Gupta P, Mandadapu H, Gourishetty S, et al. Robust transistor sizing for improved performances in digital circuits using optimization algorithms[C]//Proceedings of the 20th International Symposium on Quality Electronic Design, 2019: 85-92.

[13] WANG H B, SHI Y, GUO G, et al. Design of single-event upset self-recovery flip-flop in 40nm CMOS technology [J]. Journal of Chinese Computer Systems, 2023, 44(12): 2851-2857.

[14] Xu H, Wang J, Xu H, et al. Impact study of layout-dependent effects toward FinFET combinational standard cell optimization [J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2023, 70(2): 731-735.

[15] Jackson T. Thinking MOSFETs[J]. IEEE Transactions on Electron Devices, 2025, 72(3): 1520-1522.

[16] DONG J C, ZHANG X. Prospects of advanced integrated circuit technologies in post-moore era [J]. Review and Commentary, 2022, 1(3): 42-51.

[17] Pandey A, Garg P, Tyagi S, et al. A modified method of logical effort for FinFET circuits considering impact of fin-extension effects [C]//Proceedings of the 19th International Symposium on Quality Electronic Design, 2018: 189-195.

[18] GUO J J, ZONG J Y, ZHA P W, et al. Machine learning-based method for statistical static timing analysis of multiple input switching effects[J]. Microelectronics, 2024, 54(3): 458-467.

[19] WANG Z G, ZHANG H, LI P L. SPICE modeling and simulation of HCI degradation of CMOS devices[J]. Semiconductor Technology, 2023, 48(10): 902-910.

[20] NI W W, ZUO Y F, YAN H. Research on special floating-point computing unit for SPICE simulation[J]. Integrated Circuits and Embedded Systems, 2024, 24(2): 64-69.

[21] ZHOU R R, ZHOU W C, WANG Y. A design of carry-ahead adder based on 55nm process technology[J]. China Integrated Circuit,

2023,32(8):49-53.

[22] Gore G,Tang X,Gaillardon P E. A scalable and area-efficient configuration circuitry for semi-custom FPGA design[J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems,2023,31(8):1128-1139.

[23] JIAN Z Q. Analysis of FPGA solutions in high frequency and high speed integrated circuit chip design[J]. Application of IC,2024,41(2):320-321.

[24] ZHAO L,LI B,ZHOU Q L,et al. FPGA optimization design of full pipeline high speed AES_GCM algorithm[J]. Journal of Chinese Computer Systems,2023,44(8):1833-1841.

[25] Chinazzo A L,Wehn N,Nassif S. Enabling cell selection optimization for non-traditional CMOS topologies [C]//Proceedings of the 21st International SoC Design Conference (ISOCC),2024:139-140.

[26] Wang Y,Xin J,Fang J H,et al. Topology generic DC-model for accelerating analog circuit optimization [C]//Proceedings of the International Symposium of Electronics Design Automation (ISEDA),2023:65-70.

[27] LIN M B. Introduction to VLSI systems:a logic,circuit,and system perspective [M]. Beijing: Publishing House of Electronics Industry,2015.

[28] Zhou Y,Schuttaine J E. Latency insertion method for FinFET simulation incorporating parasitic source/drain resistances [C]//Proceedings of the IEEE 10th Electronics System-Integration Technology Conference (ESTC),2024:1-4.

[29] Gu Y,Tang C K,Li X H,et al. A structural impact study and process optimization of FinFET parasitic capacitance [J]. IEEE Transactions on Electron Devices,2025,72(1):17-23.

附中文参考文献:

[3] 孙丽莉. 集成电路标准单元版图设计技巧[J]. 集成电路应用,2022,39(1):13-15.

[13] 王海滨,侍 言,郭 刚,等. 一种基于40nm CMOS体硅工艺的抗单粒子翻转触发器设计[J]. 小型微型计算机系统,2023,44(12):2851-2857.

[16] 董俊辰,张 兴. 后摩尔时代先进集成电路技术展望[J]. 前瞻科技,2022,1(3):42-51.

[18] 郭静静,宗璟宜,查佩文,等. 基于机器学习的多输入切换效应的统计静态时序分析方法[J]. 微电子学,2024,54(3):458-467.

[19] 王正楠,张 昊,李平梁. CMOS器件热载流子注入老化的SPICE建模与仿真[J]. 半导体技术,2023,48(10):902-910.

[20] 倪文威,左芸帆,闫 浩. 面向SPICE仿真的专用浮点计算单元研究[J]. 集成电路与嵌入式系统,2024,24(2):64-69.

[21] 周冉冉,周文宸,王 永. 一种基于55nm工艺的超前进位加法器设计[J]. 中国集成电路,2023,32(8):49-53.

[23] 简震谦. 高频高速集成电路芯片设计中的FPGA解决方案分析[J]. 集成电路应用,2024,41(2):320-321.

[24] 赵 亮,李 斌,周清雷,等. 全流水线高速AES_GCM算法的FPGA优化设计[J]. 小型微型计算机系统,2023,44(8):1833-1841.

[27] 林铭波. 超大规模集成电路系统导论:逻辑、电路与系统设计[M]. 北京:电子工业出版社,2015.