NuRedact：面向低开销和安全 IP 遮蔽的非均匀 eFPGA 架构

Voktho Das、Kimia Azar、Hadi Kamali电气与计算机工程系（*ECE*），中佛罗里达大学，美国佛罗里达州奥兰多，邮编 *32816*{voktho.das ,azar,kamali}@ucf.edu

arXiv:2601.11770v1 [cs.AR] 16 Jan 2026

摘要——**尽管逻辑锁定作为应对集成电路（IC）供应链威胁的一种对策已得到广泛研究，近期研究已转向基于可重构结构的遮蔽技术，例如基于查找表和嵌入式 FPGA 的方案。这些方法虽然提高了攻击难度，却带来了显著开销，其中很大一部分并非源于实际功能对可重构性的需求，而只是人为增加复杂度以阻碍逆向工程（RE）。因此，可编程逻辑阵列往往未得到充分利用，安全性也以不成比例的代价换取。本文提出*NuRedact*，这是首个采用架构非均匀性来平衡安全性与效率的全定制 eFPGA 混淆框架。*NuRedact* 基于广泛采用的 OpenFPGA 基础设施构建，并提出了三阶段方法：（i）生成带有引脚映射不规则性的定制可编程逻辑阵列；（ii）修改 VPR，以支持由自动化的基于 Python 的优化器指导的非均匀布局；（iii）对目标 IP 模块进行感知遮蔽的重配置与映射。实验结果表明，与传统均匀架构相比，该方法的面积缩减最高可达 9× ；在保持强大韧性的同时，其效率可与基于查找表乃至晶体管级遮蔽技术相媲美。从安全性角度看，*NuRedact* 可编程逻辑阵列经评估能够抵御最先进的攻击模型，包括基于 SAT 的攻击、循环攻击及其时序变体，并在保持实用设计开销的同时展现出更强的韧性。**

索引词—**嵌入式 FPGA、硬件保护、IP 遮蔽、非均匀嵌入式 FPGA**

I. 引言

将 IC 开发的关键阶段外包给全球各方带来了规模和成本优势，但也使设计面临 IP 盗用、逆向工程（RE）和未经授权的超量生产 [1]。逻辑锁定作为一种系统化的对策应运而生 [2], ，然而，针对逻辑锁定的攻击（即去混淆攻击），包括逻辑攻击和物理攻击 [3]–[6], ，已削弱了许多锁定方案在现实威胁模型下的实际安全性，因此有必要重新评估保护原语 [2]。过去几年里，一种自然的转向是采用基于可重构结构的删改，将安全关键逻辑锥和 IP 隔离，并用可重构基底取而代之 [7]–[12],，通常采用查找表或嵌入式 FPGA（eFPGA）可编程逻辑阵列，从而使功能变得复杂难辨，且只有在受保护配置（比特流）下才能实例化。

早期的遮蔽流程已实现自动化模块选择和聚类，分别用于基于查找表的 [13] 以及基于嵌入式 FPGA 的[14] 删改。然而，这些流程将目标可编程逻辑阵列架构和布局固定下来（尤其是在基于嵌入式 FPGA 的研究中），因此留下了显著的功耗、性能与面积优化空间 [14]–[16]。过去两年，研究重点更加聚焦于可编程逻辑阵列结构和粒度[10], [11], [16]–[19]。HIPR [17] 会插入细粒度可配置模块（包括定制查找表、时序模块和互连线），分别用于实现布尔逻辑、时序逻辑和互连随机化；随后压缩配置，以降低功耗、性能与面积及比特流开销，同时保持对功能攻击和结构攻击的抵御能力。ARIANNA [18] 通过预先确定嵌入式 FPGA 参数的安全子集，并针对每个模块集群定制可编程逻辑阵列，从而将删改内容与可编程逻辑阵列的构建方式关联起来。TRAP [19] 通过采用晶体管可编程架构，将设计推进到门级以下；其开关级语义从根本上增加了攻击难度（例如，SAT 攻击 [3], [20]–[24]）。

尽管这些方法降低了开销，但它们仍依赖统一的属性配置，以满足通用性所需的布线容量、查找表结构和瓦片规整性，而非针对删改进行专门设计。因此，当删改不同的逻辑锥时，大量配置、逻辑和布线资源仍未得到充分利用 [17]–[19]。此外，比特流存储和配置触发器会带来不可忽视的硅片面积和时序开销。此类定制还可能泄露结构线索，使无预言机攻击或结构引导攻击成为可能。

在这些方法中，尽管存在开销方面的顾虑，嵌入式 FPGA 仍因其集成能力和工具链而颇具吸引力，从开源知识产权核生成器（例如 OpenFPGA [25]），到商业知识产权核（Achronix Speedcore、Flex Logix EFLX、QuickLogic），均已集成到片上系统中，并具备硅后修补能力 [25] 这一价值主张。虽然 eFPGA 的部署旨在支持未来的重新配置，但在删改（用于 IP 保护）中，这些基底被用于实现复杂性（较大的功能/搜索空间），并将显露的结构与受保护的行为解耦。这意味着，可编程逻辑阵列的非均匀性可以与目标知识产权核协同设计：删改需要的是**作为一种手段的可编程性**（安全性），而不是**产品特性**。

基于这一事实，我们提出了*NuRedact*，这是一个用于非均匀 eFPGA 脱敏的自动化框架，旨在以最低开销实现可编程性保护，同时确保安全性，并最大限度地提升集成度和工具支持。*NuRedact* 将知识产权核子组件替换为基于 OpenFPGA 生成的不规则、设计专用可编程逻辑阵列，修改 VPR 以实现非均匀布局的合法化，并纳入能够利用不规则引脚映射和定制资源的脱敏感知映射器。与 HIPR 和 TRAP 相比，*NuRedact*在开销相当的情况下，保留了主流的寄存器传输级到 OpenFPGA 自动化流程和行业工具兼容性，同时相较于基线的基于 eFPGA 的删改保护，面积最多缩减 9 倍。*NuRedact* 的贡献如下：

表 I: 脱敏方法总体比较。（■□□□□ = 低， ■■■■■ = 高）。√ = 存在， = 不存在， ∼ = 部分/间接）。

技术	粒度	开销	比特流	互连混淆	定制 可编程逻辑阵列	工具与集成	高亮
LUT-Lock [7]	LUT 混淆	■■■□□	■■■□□	✗	✗	✓	SAT 强化
Full-Lock [8]	可配置布线 + 查找表	■■■□□	■■■■□	✓	✗	✗	SAT 强化 + 证明
ALICE [15]	嵌入式 FPGA（固定架构）	■■■■■	■■■■□	✗	✗	✓	端到端流程
SheLL [10]	嵌入式 FPGA（逻辑块缩小）	■■■□□	■■■□□	∼	✓	✓	↓∼55% 对比 ALICE [15]
HIPR [17]	查找表 + 时序逻辑 + 互连	■■□□□	■■□□□	✓	✗	✓	安全感知压缩
ARIANNA [18]	嵌入式 FPGA（定制、统一瓦片）	■■□□□	■■■□□	∼	✓	✓	3.3× 比固定架构更低
SOAR [12]	嵌入式 FPGA +监控	■■■■■	■■■■□	✓	✗	✓	动态密钥
TRAP [19]	晶体管级结构	■□□□□	■■■□□	✗	✗	∼	SAT 强化
NuRedact（拟议）嵌入式 FPGA（非均匀）		■■□□□	■■■□□	∼	✓	✓	利用率不足较低 黑盒可学习性较低 相比固定架构，尺寸最多缩小 9 倍

评级为定性评估，依据先前研究报告的结果综合得出。

(i) 一种自动化框架，用于生成设计专用的**非均匀** 嵌入式 FPGA 结构，在保留被脱敏逻辑完整功能的同时降低面积开销。

(ii) 一个基于 Python 的软件包，用于在 SkyWater 130nm OpenFPGA 架构上进行布局变换和 VPR 架构定制，以生成兼容工具链的可编程逻辑阵列。

（iii）一种以安全为导向且兼顾开销的选择性删改方法，以不规则、非均匀的织物替换关键逻辑，从而最大限度地降低开销。

（iv）使用 Cadence Genus 进行综合，并使用 SynopsysDesign Compiler(DC)生成便于攻击的网表以评估开销（通过 IcySAT 展开和 KC2 SAT工具链进行评估）。

II. 背景与相关工作

A. 通过删改实现硬件*IP*保护

现有删改技术可分为三大类：（i）细粒度LUT删改，（ii）粗粒度eFPGA删改，以及（iii）亚门级晶体管层删改：

（i）细粒度*LUT*删改。这些技术通过用小型查找表替换选定的逻辑门/逻辑锥来删改逻辑 [7], [13], [17],[26], [27]。这些技术通过压缩查找表 [7], 、采用抗攻击插入策略 [7], [13], [17], 以及比特流压缩 [17]来控制开销。这些技术的优点是便于布局，且集成到标准网表所需的工作量适中。但若压缩或布局不当，结构可能会泄露 [17], [27]。

(ii) 细粒度查找表+布线删改。包括 HIPR 在内的多项研究 [17],通过配置级联交织多路复用器，扩展了基于查找表的删改方案，使其同时实现查找表功能和布线交叉开关 [8], [28]。该设计与 FPGA架构 高度相似，其中部分配置用于启用逻辑运算，其余部分则支持布线恢复。尽管该方法具有较高的鲁棒性，但仍容易受到基于链接预测的机器学习攻击 [29]。

（iii）粗粒度 *eFPGA* 删改。寄存器传输级模块（或 IP）被移至嵌入式 FPGA 宏之后，早期的端到端流程会自动确定要删改的内容 [15]。较新的框架则会在预筛选以确保配置安全后，针对每个集群定制可编程逻辑阵列（定制的 K/N/IO/逻辑块网格），从而提高利用率并降低开销 [18]。这些技术得益于已揭示结构与隐藏行为之间的强解耦，以及成熟的工具流程（OpenFPGA/VPR）带来的全面自动化。然而，它们往往为删改配置了过多资源，浪费逻辑/布线并增加开销。此外，配置寄存器和 I/O 引脚可能会为攻击推断提供线索，导致功能/比特流泄露 [30]–[33]。

(iv) 亚门级晶体管级删改。开关级可编程逻辑阵列有意打破门级假设，以降低开销 [19]。过去已有类似方法仅用于基于布线的混淆 [34], ，而较新的方法则转向晶体管级的全可配置基底。这些方法消除了层次化LUT线索。然而，与 LUT/eFPGA流程 相比，其工具链（自动化程度）并不那么完善易用。

表 I 展示了这三类中关键研究的详细比较。*NuRedact* 保留了 eFPGA混淆 的集成性和自动化能力，但摒弃了均匀性。通过将非均匀架构与目标逻辑锥进行协同设计，它在保留删改安全模型的同时，大幅减少了利用率不足和面积开销，恰好弥补了上述现有技术所揭示的空白。

B. 定制化嵌入式 *FPGA* 设计框架

eFPGA 删改流程包含：(i) 一种架构描述（例如 VTR/VPR XML），用于定义网格布局、瓦片、逻辑集群、布线等；(ii) 一种可编程逻辑阵列生成器（例如 OpenFPGA），用于将架构原语与寄存器传输级及比特流工具绑定；以及 (iii) 一条专用集成电路集成路径，使用供应商的编译器（例如 Cadence Genus）编译嵌入式 FPGA 宏，以完成时序、平面规划、可测试性设计等工作。 [25], [35]。目前已有多种开源和商业框架用于将 eFPGA 集成到专用集成电路中，即基于 eFPGA 的片上系统（SoC）；其中，VTR/VPR [35] +和 OpenFPGA [25] 被广泛用于原型设计。这些工具链为配置提供了关键选项，包括逻辑聚类（查找表尺寸和可分解性、进位链及硬核模块）、布线拓扑（例如通道宽度、开关盒类型和交叉开关拓扑）、时钟以及配置协议（基于帧或基于随机存取存储器） [10], [25]。

关键在于，尽管可编程逻辑阵列通常被描述为完全可平铺且均匀，但 VTR/VPR 和 OpenFPGA 原生

支持异质性和不规则性 [25], [35], , 包括混合瓦片类型、使用不规则引脚映射、以不对称方式删减连接/开关盒，以及改变通道宽度。通过引脚等价类、布局区域和显式资源约束，可以保持合法性；因此，即使可编程逻辑阵列偏离棋盘格（网格模型） [35], [36], 标准的打包/布局/布线流程仍然正确。对于删改而言，这些能力之所以重要，是因为其目标并非赋予终端用户灵活性，而是在较低硅成本和最小结构规则性的条件下，构建庞大的功能/搜索空间。*NuRedact*通过围绕目标逻辑锥对可编程逻辑阵列进行协同设计，采用不规则平铺、引脚映射不对称性和选择性逻辑单元删减来实现删改；随后通过显式的布局器/布线器约束和绑定元数据强制确保合法性，从而在保持安全性的同时降低开销，并与 OpenFPGA/VTR 及标准 ASIC流程实现无缝集成。

*C. eFPGA*删改保护的攻击与安全分析

针对删改保护的攻击旨在（i）恢复部分或全部功能 [30], [32] ，或（ii）缩小搜索空间，直到剩余配置位可被求解。 [16], [33]。对于基于 LUT 的方案，早期的随机 LUT 插入很快就被 SAT 去混淆攻破 [37]。后来的启发式方法虽然提高了破解难度，却泄露了结构信息；基于机器学习的攻击会利用缺失的 k-割来推断 LUT布局/密钥 [29]。对于基于 eFPGA 的删改保护，可编程逻辑阵列的均匀性削弱了直接结构推断的效果，但黑盒功能攻击仍然可行，例如 FuncTeller 会查询 eFPGA 的输入/输出，识别输出集最小项，将其扩展为主蕴含项，并综合出近似网表 [30]。与此同时，SAT 推导（例如时序/循环 SAT，如展开）在处理好循环后，可以压缩有效配置空间，从而实现预言机引导的恢复 [23], [24], [38]。近年来，MANTIS 等数据驱动方法无需显式 SAT 模型，就能根据观测到的输入/输出行为学习近似位流，这表明攻击面并不限于传统的逻辑级形式化[32]。这些趋势表明，可以通过架构调节参数降低攻击成功率（NuRedact 针对的正是这一点）：（i）限制机器学习模型和无预言机方法所利用的结构规则性；（ii）避免低利用率、过-已配置的嵌入式 FPGA 架构。

III. 威胁模型

与以往的逻辑隐去研究一致，资产（攻击目标）包括：(i) 隐去逻辑锥的功能，(ii) 实现这些功能的嵌入式 FPGA 配置（比特流），以及 (iii) 架构位与配置地址之间的映射。攻击者的目标是通过恢复正确（或近似）的配置，或重构被隐去的功能，来复制受保护逻辑。攻击者掌握 ASIC 的隐去门级网表，并可对正常运行的设备（预言机）执行输入输出查询。我们评估的攻击面包括预言机引导的 SAT 系列、黑盒功能恢复，以及无预言机／结构引导攻击。我们假设比特流不会以明文形式提供，且配置接口位于系统安全边界之内。此外，针对比特流加载器的侵入式物理提取和高级侧信道攻击虽很重要，但为与以往逻辑隐去研究保持一致，本文不予讨论。

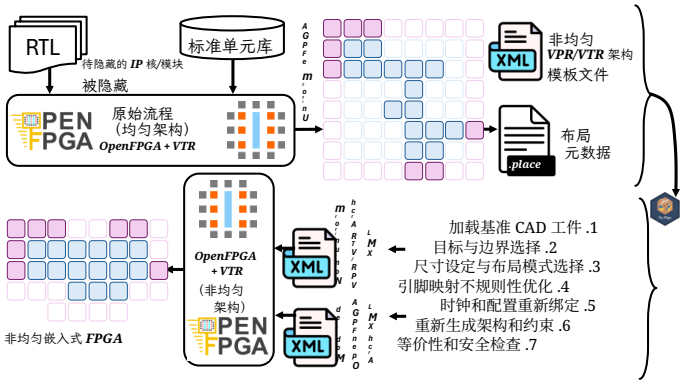


图 1： *NuRedact* 框架：利用均匀流程中的布局数据，并在 OpenFPGA 中集成基于 Python 的 VPR/VTR XML 定制功能，自动生成定制的非均匀嵌入式 FPGA 架构。

加载器的攻击也很重要，但为与以往的逻辑隐去研究保持一致，本文不予讨论。

IV. 提出的删改方案：*NuRedact*

NuRedact 是一种全自动流程，可合成针对特定设计的、首创的非均匀嵌入式 FPGA 架构，用于删改。为确保自动化可靠（让所有环节都运行在广泛使用的框架上，例如 OpenFPGA 和 VPR/VTR），其核心思路是从基线 VPR/OpenFPGA 布局布线（基线均匀运行）中提取结构化元数据，然后重新生成定制布局，在保留功能和 CAD 合法性的同时，删减未充分利用的逻辑/布线，并注入架构不规则性（引脚映射、逻辑单元删减、区域分段）。最终得到的可编程逻辑阵列占用面积紧凑，与删改后的锥体占用面积相匹配，可与未修改的工具链集成，并以低得多的开销维持安全性。图 1 展示了*NuRedact* 的整体实现流程，其中突出显示了用于布局变换的定制软件包、定制化 VPR/VTR 架构生成以及 OpenFPGA 运行的集成。

*A. Uni*可编程逻辑阵列分析与逻辑利用率补充章节 部分

NuRedact 从规则（均匀）的嵌入式 FPGA 架构开始处理，该架构由规则排列的同质逻辑 tile 和 I/O 引脚组成。设计通过标准的 OpenFPGA+VPR 流程进行打包、布局和布线，生成规范的构件（包括记录所有已布局逻辑块和输入/输出单元空间坐标的 .place 文件）。通过处理 .place 日志，*NuRedact* 可提取利用率图（已占用的瓦片和输入/输出）、布线图以及各条网络的时序关键性。此分析可消除未使用的逻辑和输入/输出资源，并启动定制的非均匀布局，仅保留实际使用的瓦片和输入/输出。

B. 非均匀布局的自动生成

根据布局数据，NuRedact 中实现了一个基于 Python 的布局转换框架，用于构建定制的非均匀架构（依据 .place 日志中的结构化元数据）。该框架基于两项输入生成新的架构 XML 文件，并更新布局和布线配置：(i) 来自 .place 日志的结构化元数据；(ii) 参数化的 VPR 架构模板 XML 文件。

算法1 *NuRedact*：非均匀结构生成

输入： P：布局文件，　A_{temp}：模板架构
输出： A_{nu}：非均匀架构
$C_{io} \leftarrow$IO容量来自 A_{temp}
$(N_{clb},\ N_{io}) \leftarrow$ 解析(P)
$B_{io} \leftarrow \left\lceil \frac{N_{io}}{C_{io}} \right\rceil$
$(W,\ H) \leftarrow$最小值$_{w,\ h}(w+2)(h+2)$　使得 $wh \geq N_{clb},\ 2(w+h) \geq B_{io}$
放置 N_{clb} 可配置逻辑块 $\rightarrow \mathcal{C}$
边界IO $\rightarrow \mathcal{I}$
当 $|\mathcal{I}| < B_{io}$ **do**
　$\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{P}$
结束当
$A_{nu} \leftarrow \mathcal{C} \cup \mathcal{I}$
导出 $A_{nu} \rightarrow$ VPR XML

该配置反映了精简且设计专用的布局，同时满足 VPR 的输入要求，如算法1所示。

与使用自动_布局来定义瓦片排列的传统 VPR 架构不同，自定义的非均匀架构依赖于固定布局规范。这使得可以显式定义瓦片位置，从而精确控制逻辑块和 I/O 引脚的空间组织。固定布局中的总体阵列大小并非静态，而是根据原始布局中已使用瓦片的空间范围动态确定。因此，最终可编程逻辑阵列的尺寸直接反映设计的实际资源需求。我们的框架支持两种布局策略：

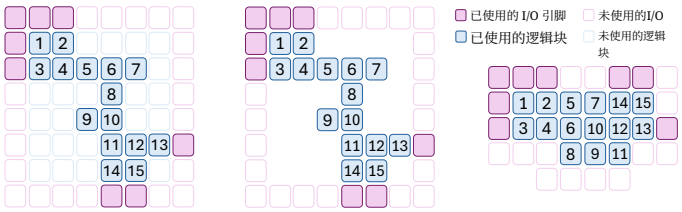
（i）保守缩减：保留所有已使用逻辑块的原始空间坐标，并移除所有未使用的瓦片。仅保留所需数量的 I/O 瓦片，且每侧至少保留一个 I/O 瓦片，以维持布线可访问性。这种方法在减少资源的同时，确保对布线和时序的影响最小。图2(b)展示了该策略的一个示例。

(ii) 布局紧凑化：通过将所有已使用的逻辑块重新定位到连续且紧密排列的区域，实现激进的紧凑化。此重新定位可缩小可编程逻辑阵列的整体尺寸，并改善空间局部性。根据与已使用逻辑块的邻接关系选择 I/O 瓦片；在适用时，通过调整每个逻辑块的 I/O 容量来减少 I/O 瓦片的数量。图 2(c) 展示了这一策略的示例。为了支持紧凑布局中的灵活布线，*NuRedact*会在定制VPR XML中添加一个额外的布线段。具体而言，在默认的长布线段（L4）之外，还引入了一个短布线段（L1）。L4 跨越四个逻辑块，便于在可编程逻辑阵列中进行较长距离的通信；而新增的 L1 布线段长度仅为一个逻辑块。这种更精细的布线可改善局部连通性，并且

这对于紧密排列的非均匀布局尤其有利，因为其中短互连线占主导。结合使用 L1 和 L4 布线资源可提升可布线性，并缓解因逻辑块间距不规则而可能产生的拥塞。

C. 可编程逻辑阵列重生成与流程兼容性

转换脚本生成的自定义架构文件通过再次运行 OpenFPGA 流程，用于重新生成可编程逻辑阵列。此阶段生成非均匀嵌入式 FPGA 架构，包括布线基础设施和



(a) 均匀嵌入式 FPGA 架构（利用不足）　(b) 非均匀嵌入式 FPGA（保守缩减）　(c) 非均匀嵌入式 FPGA（激进紧凑化）

图 2： *NuRedact* 中的均匀与非均匀嵌入式 FPGA：(b) 保守删改与 (c) 激进布局紧凑化。

配置比特流生成支持。重新生成的架构仍与原始 RTL 设计和综合结果兼容。该流程保留所有逻辑映射并确保功能一致，同时实现更小面积的设计。集成到 OpenFPGA 的基于任务的基础设施后，整个流程——从均匀架构运行开始，到生成综合后的非均匀架构结束——都可以自动执行，无需人工干预。*NuRedact*可用于构建与逻辑无关、通用且紧密匹配实际设计需求的非均匀嵌入式 FPGA 架构。通过消除未使用的逻辑块并紧凑布局，生成的架构在不牺牲正确性或工具兼容性的前提下，展现出更高的空间效率和更低的开销。

V. 使用*NuRedact* 框架进行安全 IP 隐去

从删改角度看，*NuRedact* 引入了受空间约束的 IP 内部隐去，使得能够以较低的外部可观测性选择性地隐去关键子组件，例如加密核的算术模块、控制逻辑或决策单元。由于 *NuRedact* 最大化了每个逻辑块的利用率（通过删减逻辑单元），逻辑与输入/输出的比率将大幅提高（在更小的占用面积内实现更密集的逻辑，并减少 I/O 引脚）。因此，这种非均匀删改结合引脚映射不对称性，可最大限度降低算法攻击（例如基于 SAT 的攻击）学习黑盒的能力。我们的实验（见第 VI-B 节）表明，在更小的占用面积内实现密集逻辑，可使每单位比特流大小对应的 SAT 变量增加一个数量级，展现了在较低 PPA 开销下实现的安全性提升。

此外，在 *NuRedact* 中，ASIC 与 eFPGA 之间的边界通过一种简单的多目标策略确定：在满足时序裕量和布局规划约束的前提下，尽量缩小接口宽度（输入/输出切割），并降低暴露的可控性/可观测性。实践中，我们根据以下因素对候选切割进行排序：(i) 切割信号数量和翻转/观测指标，(ii) 逻辑的独特性/稀有程度，以及 (iii) 最关键网络的绕行预算。通过引脚映射不对称性（置换等效引脚以打破端口规则性）以及精简逻辑/布线，引入架构不规则性，进一步增强安全性，消除可预测的架构模式。与方形均匀覆盖层相比，这种方法 (i) 去除了利用不足的硅片，(ii) 减少了有助于无预言机推断的视觉和图结构线索，(iii) 提高了每个逻辑单元的配置熵，同时仍可使用标准 OpenFPGA/VTR 和签核流程。

表 II： 所选基准电路的规格。

基准	# 模块	# 输入	# 输出	已隐去的模块
GPS	12	6-128	1-256	Cacode
RISC-V	18	3-130	1-79	CTRL 仲裁器 加载/存储
DES3	11	6-240	4-64	替换盒 8_
AES	9	10-128	8-128	AES_ShR
加法器	2	3-4	2-3	全加器阵列_ 半加器阵列_
GCD	8	8-45	1-18	比较器

VI. 实验结果与分析

为评估所提出的 *NuRedact*框架的有效性，我们在一组基准测试上开展了实验，其中包括基于 RISC-V 的 PULPino 片上系统，以及表 II 所列的其他 IP核。对于每个基准测试，如图所示，我们使用 OpenFPGA 工具链[1]选择特定模块作为基于 eFPGA 的删改保护目标。*NuRedact* 使用 Python 实现，并借助 PyVerilog 进行词法解析和非均匀嵌入式 FPGA 的生成，所有这些功能均集成到 OpenFPGA 流程中。我们使用采用 SkyWater 130nm 工艺库的 Cadence Genus 21.18 执行逻辑综合，以评估面积开销；实验在一台配备 Intel Xeon E5-2640 v4 2.4 GHz 处理器（40 核、48 GB 随机存取存储器）的服务器上进行。为评估脱敏织构的安全性，我们使用 Synopsys DC 在 Nangate 15 纳米工艺节点上生成了兼容 IcySAT 和 KC2 SAT 攻击工具的网表。随后，在配备 Intel® CoreTM i7-14700 处理器（28 核、32 GB 随机存取存储器）的系统上通过 Ubuntu 环境，对生成的非均匀嵌入式 FPGA 架构发起攻击。

A. 硬件开销分析

NuRedact 框架的首要目标是在保留功能并确保能够抵御攻击的稳健性的同时，尽可能缩小 eFPGA 可编程逻辑阵列的面积。对于每个基准，*NuRedact* 流程都会生成均匀（基线）和非均匀架构。逻辑结构布局由 VPR 生成，使用定制的 VPR 架构文件以及

表 III： 与传统均匀架构相比，通过*NuRedact* 框架生成的嵌入式FPGA结构的面积效率。

IP	模块	可编程逻辑阵列 # 总计 # 已使用 Size	瓦片	瓦片	面积(μm^2) 均匀	非均匀	△面积 (%)
GPS	Cacode	3x3	9	7	513,361	357,251	-30.41
AES	[illegible]	2x2	4	3	212,716	158,113	-25.67
序列_组合 时序逻辑_组合		3x3	9	5	560,845	257,653.5	-54.06
RISC-V	CTRL	3x3	9	5	529,113	306,957	-41.99
	仲裁器	6x6	36	5	2026 年,867	353,273.5	-82.57
	加载/存储	1x1	1	1	64,195	57,833	-9.91
Logic7	Logic7	3x3	9	7	513,361	357,251	-30.41
加法器 FA_阵列		1x1	1	1	64,195	57,833	-9.91
DES3	SBOX_8	1x1	1	1	64,195	57,833	-9.91

负值 △面积（%）的值表示相对于均匀 abric.

[1]在这些基准测试中，我们选取较小的范围，以直观展示*NuRedact* 在各逻辑阵列上的紧凑化效果。

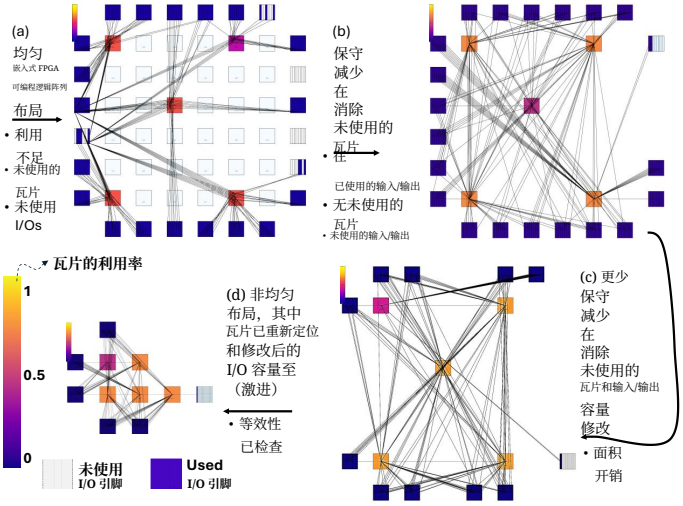


图 3： 使用*NuRedact* 框架（分别采用保守和激进策略）生成的 RISC-V 基准测试中仲裁器模块的均匀与非均匀嵌入式 FPGA 逻辑阵列布局对比。

目标模块的 BLIF表示。图 3 和图 4 展示了两个 RISC-V 模块从均匀架构到紧凑非均匀架构的变换流程，这些模块均取自 OpenFPGA 框架。仲裁器（图 3(a)）的架构为 6x6；移除未使用的可配置逻辑块后，得到缩减后的非均匀布局（图 3(b)）；随后通过保守缩减进一步优化（图 3(c)），剪除未使用的可配置逻辑块并调整输入/输出容量。接着进行紧凑布局生成（图 3(d)），此时输入/输出容量增加，已使用的逻辑块被重新定位到连续区域，从而获得更紧凑的可编程逻辑阵列。这一流程将仲裁器的占用面积从6×6 个瓦片缩减至五瓦片设计（面积缩减 ~86% 或 7.14 倍），同时保持标准工具流程不变。

CTRL 也呈现相同趋势（图 4），不过由于其逻辑占用面积更贴合，紧凑化比率较低。OpenFPGA 生成的热力图证实，*NuRedact* 提高了每个逻辑块的利用率：对于仲裁器，平均逻辑块利用率从 0.68（均匀布局）上升至 0.94（激进紧凑化）。更高的利用率减少了闲置配置位，并提高了单位面积的配置熵密度，从而减少结构线索并限制黑盒可学习性，同时保持 CAD 合法性以及与 OpenFPGA/VTR 的兼容性。

我们使用 Cadence Genus，在相同的单元库和时序约束下，对均匀和*NuRedact* 非均匀逻辑阵列进行综合，以比较硅片面积。表 III 总结了所有已隐去模块的结果。例如，对于 Seq_Comb（包含 5 个已使用逻辑块的 3×3 逻辑阵列），*NuRedact* 将面积缩减了 54.06%（2.18 倍）；对于 RISC-V 仲裁器（基准面积为 6×6），缩减幅度达到 82.57%[2]（5.73 倍），凸显了根据实际利用率调整逻辑阵列占用面积的效率。

表 IV 比较了两种非均匀模式——保守缩减与布局紧凑化，同时扫描每个逻辑块的 I/O 容量（例如 4、8、16、32；Arbiter 为 48）。为了

[2] 根据 OpenFPGA 布局计数估算的每瓦片面积缩减为 ≈ 86%，而 Cadence Genus 报告的综合后面积缩减为 82.57%；两者结果接近，验证了逐逻辑块比较可作为硅片面积的快速近似指标。

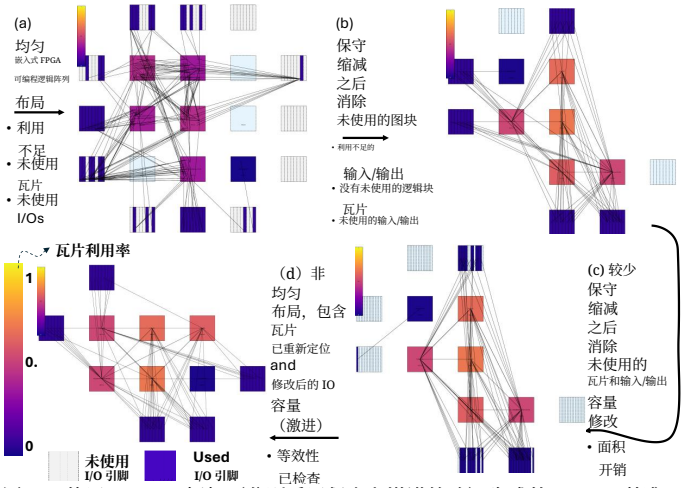


图 4： 使用*NuRedact*框架（分别采用保守和激进策略）生成的 RISC-V 基准 CTRL 模块均匀与非均匀嵌入式 FPGA 逻辑阵列布局对比

保持可布线性，即使需求低于四个，我们仍在可编程逻辑阵列的每条边上至少保留一个 I/O 瓦片。总体趋势一致：随着每个逻辑块的 I/O 容量增加，*NuRedact* 的布局紧凑化相比保守缩减可实现更大的面积节省。例如，对于 Arbiter 这类规模较大、输入/输出数量较多的模块，激进模式下，紧凑化相较保守模式可减少 37.88% 的面积；相较均匀模式，总计减少 89.17%（缩减 9.23 倍）。在所有基准测试中，布局紧凑化生成的可编程逻辑阵列最为紧凑（面积最小），而当时序或可布线性要求严格时，保守缩减则提供了风险较低的备用方案。

B. 去混淆分析

在安全分析中，我们将每个目标模块分别删改到两种可编程逻辑阵列上：(i) 标准 SkyWater-130 OpenFPGA 架构，以及 (ii) 由*NuRedact*生成的非均匀 VPR 架构。两种可编程逻辑阵列都采用扫描链配置触发器和可分解的 4 输入查找表。与先前最先进的基于嵌入式 FPGA（以及其他基于可重构技术）的研究[17]–[19], [31],类似，我们对*NuRedact* 的安全分析将仅关注派生 SAT 攻击。这是因为基于嵌入式 FPGA 的删改本质上能够抵御其他类别的去混淆攻击，

表 IV： *NuArch*中的面积优化：保守与激进。

模块	#输入/输出/瓦片	保守 #I/O 瓦片	保守 Area (μm^2)	激进 #所需输入/输出（已分配）	激进 Area (μ米^2)	Δ面积 (%)
时序逻辑_组合逻辑	4	6	264,708	6	255,618	-3.43
	8（默认值）	6	270,975	3(4)	257,653.5	-4.92
	16	6	280,724	2(4)	264,199	-5.89
CTRL	4	10	315,753.5	10	307,256.5	-2.69
	8（默认值）	7	319,507	5	306,957	-3.93
	16	7	331,092.5	2(4)	311,303	-5.98
仲裁器	8（默认值）21		384,266	21	353,273.5	-8.06
	16	11	381 ,784.5	11	321,403	-15.82
	32	6	410,700	6	325,971	-20.63
	48	6	527,432.5	4	327,645.5	-37.88
Logic7	4	10	363,347.5	6	355,217.5	-2.24
	8（默认值）	10	373,705.5	4	357,251	-4.40
	16	10	389,769.5	2(4)	363,775.55	-6.67

Δ面积（%）为负值表示面积缩减。

表 V： 通过*NuRedact* 框架生成的非均匀嵌入式FPGA结构上的SAT攻击结果。

设计	比特流	展开因子 #	子句#	变量	时间 (s)	密钥 (S/F)
HA_阵列	915	98	6165189	2318300	8102.2	S
CTRL	8215	434	-	-	TO	F
FA_阵列	937	254	19595603	7327291	40818.5	S
Cacode	9582	534	-	-	TO	F
加载/存储	937	254	-	-	TO	F
比较器	861	303	-	-	TO	F
高级加密标准_逻辑右移	2252	813	-	-	TO	F
仲裁器	8278	-	-	-	TO	F
替换盒 8_	2001	-	-	-	TO	F
Logic7	4787	-	-	-	TO	F

例如结构化方法或基于机器学习的方法 [31], [33]。为进行 SAT 式分析（包括循环和时序逻辑），我们通过将所有配置触发器暴露为主密钥输入，将删改后的可编程逻辑阵列转换为 .bench 格式。配置触发器通过 prog_clk 驱动并串接成一条扫描链（我们通过从 scan_in_head 到 scan_in_tail 的深度优先遍历，恢复正确的比特顺序），随后通过与 OpenFPGA 比特流进行重新仿真来验证映射。对于时序/循环行为，我们使用 IcySAT 展开复杂的组合环路，并通过将密钥输入固定为 OpenFPGA 比特流中的真实配置来实例化预言机。然后，我们在暴露了密钥的网表上运行 KC2，以评估功能恢复情况。（所有运行均采用相同的时间/内存预算；均匀与非均匀方案在相同条件下进行评估。）

表 V 总结了 NuRedacted 可编程逻辑阵列上的 SAT 求解结果。对于一个小型设计（2 位加法器，比特流大小为 915），IcySAT 展开了 98 个循环，KC2 在 2+ 小时内恢复了密钥。与其均匀版本相比，非均匀可编程逻辑阵列在 NuRedacted 模块上引入了 ≥10× 个 SAT 变量，大幅增加了求解时间，这是因为结构线索和黑盒可学习性有所降低。对于较大的设计和比特流（例如 RISC-V CTRL 和 GPS Cacode），IcySAT 经过很长时间后完成了展开，但 KC2 未能在可用的随机存取存储器和时间预算内恢复密钥。

VII. 结论

本文介绍了*NuRedact*，这是一种自动化的基于 eFPGA 的删改保护流程，它以针对特定设计的非均匀嵌入式 FPGA 结构替换安全关键逻辑。*NuRedact* 通过引入引脚映射不对称性、选择性逻辑单元删减、局部布线和紧凑固定布局，解决了 eFPGA 利用率不足的问题，同时兼容 OpenFPGA/VTR 和标准签核，符合 CAD 规范。在 RISC-V 及其他基准测试中，*NuRedact* 显著提升了效率：面积最多减少 9 倍，每个 tile 的利用率平均提升 38%，有助于提高单位面积的配置熵密度，从而减少结构线索并降低黑盒可学习性。在（循环和时序）攻击评估中，极小的结构仍可被求解，但包含 ≥5 个逻辑块的非均匀结构能够有效抵御去混淆攻击（待确定变量增加了 10 倍）。

参考文献

[1] M. Rostami、F. Koushanfar 和 R. Karri，《硬件安全入门：模型、方法与度量指标》，*IEEE* 会刊，第 102 卷，第 8 期，第 1283–1295 页，2014 年。[2] H. M. Kamali、K. Z. Azar、F. Farahmandi 和 M. Tehranipoor，《逻辑锁定的进展：过去、现在与展望》，*Cryptology ePrint Archive*，2022 年。[3] P. Subramanyan、S. Ray 和 S. Malik，《评估逻辑加密算法的安全性》，载于*2015* 年 *IEEE* 硬件导向安全与信任国际研讨会（*HOST*）。IEEE，2015 年，第 137–143 页。[4] K. Z. Azar、H. M. Kamali、H. Homayoun 和 A. Sasan，《SMT 攻击：针对混淆电路的新一代攻击，其能力和性能超越 SAT 攻击》，*IACR* 密码硬件与嵌入式系统汇刊，第 97–122 页，2019 年。[5] A. Barenghi、L. Breveglieri、I. Koren 和 D. Naccache，《密码设备的故障注入攻击：理论、实践与对策》，*IEEE* 会刊，第 100 卷，第 11 期，第 3056–3076 页，2012 年。[6] P. Kocher、J. Jaffe 和 B. Jun，《差分功耗分析》，载于国际密码学年会。Springer，1999 年，第 388–397 页。[7] H. M. Kamali、K. Z. Azar、K. Gaj、H. Homayoun 和 A. Sasan，《Lut-lock：一种新颖的基于 LUT 的逻辑混淆技术，用于 FPGA 比特流和专用集成电路硬件保护》，载于*2018 IEEE* 计算机学会年度 *VLSI* 研讨会（*ISVLSI*）。IEEE，2018 年，第 405–410 页。[8] H. M. Kamali、K. Z. Azar、H. Homayoun 和 A. Sasan，《Full-lock：利用完全可配置逻辑与布线模块生成用于混淆电路的困难 SAT 实例分布》，载于第 *56* 届年度设计自动化会议论文集（*2019*），2019 年，第 1–6 页。[9] L. Collini、B. Tan、C. Pilato 和 R. Karri，《用于硬件 IP 保护的可重构逻辑：机遇与挑战》，载于第 *41* 届 *IEEE/ACM* 计算机辅助设计国际会议论文集，2022 年，第 1–7 页。[10] H. M. Kamali、K. Z. Azar、F. Farahmandi 和 M. Tehranipoor，《SheLL：缩减用于逻辑锁定的嵌入式 FPGA 结构》，载于*2023* 年欧洲设计、自动化与测试会议暨展览会（*DATE*）。IEEE，2023 年，第 1–6 页。[11] Z. U. Abideen、S. Gokulanathan、M. J. Aljafar 和 S. Pagliarini，《受 FPGA 启发的混淆技术综述》，*ACM* 计算评论，第 56 卷，第 12 期，第 1–35 页，2024 年。[12] V. Das、K. Azar 和 H. Kamali，《SOAR：安全一次，运行时自适应的基于 eFPGA 的 IP 删改保护》，*IEEE Access*，2025 年。[13] R. Guo、M. S. Rahman、H. M. Kamali、F. Rahman、F. Farahmandi 和 M. Tehranipoor，《Evolute：基于查找表的细粒度 IP 删改保护评估》，载于*2023* 年欧洲设计、自动化与测试会议暨展览会（*DATE*）。IEEE，2023 年，第 1–6 页。[14] J. Bhandari、A. K. T. Moosa、B. Tan、C. Pilato、G. Gore、X. Tang、S. Temple、P.-E. Gaillardon 和 R. Karri，《探索基于 eFPGA 的 IP 删改保护》，载于*2021* 年 *IEEE/ACM* 计算机辅助设计国际会议（*ICCAD*）。IEEE，2021 年，第 1–9 页。[15] C. M. Tomajoli、L. Collini、J. Bhandari、A. K. T. Moosa、B. Tan、X. Tang、P.-E. Gaillardon、R. Karri 和 C. Pilato，《ALICE：用于 eFPGA 删改保护的自动设计流程》，载于第 *59* 届 *ACM/IEEE* 设计自动化会议论文集，2022 年，第 781–786 页。[16] J. Bhandari、A. K. T. Moosa、B. Tan、C. Pilato、G. Gore、X. Tang、S. Temple、P.-E. Gaillardon 和 R. Karri，《并非所有结构都生而平等：探索用于 IP 删改保护的 eFPGA 参数》，*IEEE* 超大规模集成电路系统汇刊，第 31 卷，第 10 期，第 1459–1471 页，2023 年。[17] A. Dasgupta、S. Paria 和 S. Bhunia，《HIPR：通过低开销细粒度删改保护实现硬件 IP 保护》，*IACR* 密码硬件与嵌入式系统汇刊，第 2025 卷，第 3 期，第 781–805 页，2025 年。[18] L. Collini、J. Bhandari、C. Muscari Tomajoli、A. Moosa、B. Tan、X. Tang、P.-E. Gaillardon、R. Karri 和 C. Pilato，《ARIANNA：用于结构定制和 eFPGA 删改保护的自动设计流程》，*ACM* 电子系统设计自动化汇刊，2025 年。[19] A. Fowler、S. Mohammed、M. Shihab、T. Broadfoot、P. Beerel、C. Sechen 和 Y. Makris，《针对 SAT 的陷阱：晶体管级可编程结构对基于可满足性攻击的免疫性》，*IACR Transactions* 《密码硬件与嵌入式系统》，第 2025 卷，第 2 期，第 579–603 页，2025 年。[20] M. El Massad、S. Garg 和 M. V. Tripunitara，《针对集成电路伪装的SAT攻击：影响与潜在对策》，IEEE集成电路与系统计算机辅助设计汇刊，第 39 卷，第 8 期，第 1577–1590 页，2019 年。[21] K. Shamsi、M. Li、T. Meade、Z. Zhao、D. Z. Pan 和 Y. Jin，《AppSAT：近似去混淆集成电路》，载于 2017 年 IEEE 国际硬件安全与信任专题研讨会（HOST），IEEE，2017 年，第 95–100 页。[22] Y. Shen 和 H. Zhou，《Double Dip：重新评估逻辑加密算法的安全性》，载于《2017 年大湖区超大规模集成电路研讨会论文集》，2017 年，第 179–184 页。[23] H. Zhou、R. Jiang 和 S. Kong，《CycSAT：针对循环逻辑加密的基于 SAT 的攻击》，载于《2017 年 IEEE/ACM 计算机辅助设计国际会议（ICCAD）论文集》，IEEE，2017 年，第 49–56 页。[24] K. Shamsi、D. Z. Pan 和 Y. Jin，《IcySAT：针对循环锁定电路的改进型基于 SAT 的攻击》，载于《2019 年 IEEE/ACM 计算机辅助设计国际会议（ICCAD）论文集》，IEEE，2019 年，第 1–7 页。[25] X. Tang、E. Giacomin、A. Alacchi、B. Chauviere 和 P.-E. Gaillardon，《OpenFPGA：支持快速原型设计的可定制 FPGA 开源框架》，载于《2019 年第 29 届国际现场可编程逻辑与应用会议（FPL）》，IEEE，2019 年，第 367–374 页。[26] A. Baumgarten、A. Tyagi 和 J. Zambreno，《利用可重构逻辑屏障防止集成电路盗版》，《IEEE 计算机设计与测试》，第 27 卷，第 1 期，第 66–75 页，2010 年。[27] G. Kolhe、S. M. PD、S. Rafatirad、H. Mahmoodi、A. Sasan 和 H. Homayoun，《基于定制查找表的混淆》，载于《2019 年大湖区超大规模集成电路研讨会论文集》，2019 年，第 477–482 页。[28] H. M. Kamali、K. Z. Azar、H. Homayoun 和 A. Sasan，《InterLock：一种互相关逻辑与布线锁定方法》，载于《第 39 届计算机辅助设计国际会议论文集》，2020 年，第 1–9 页。[29] L. Alrahis、S. Patnaik、M. A. Hanif、M. Shafique 和 O. Sinanoglu，《Untangle：利用基于图神经网络的链路预测解锁布线与逻辑混淆》，载于《2021 年 IEEE/ACM 计算机辅助设计国际会议（ICCAD）》，IEEE，2021 年，第 1–9 页。[30] Z. Han、M. Shayan、A. Dixit、M. Shihab、Y. Makris 和 J. Rajendran，《{FuncTeller}：{eFPGA} 隐藏功能的效果如何？》，载于第 32 届 USENIX 安全研讨会，2023 年，第 5809–5826 页。[31] P. Karmakar、M. Bharani 和 C. Karfa，《评估大规模基于 eFPGA 的硬件删改保护的鲁棒性》，载于《2024 年第 37 届超大规模集成电路设计国际会议暨 2024 年第 23 届嵌入式系统国际会议（VLSID）》，IEEE，2024 年，第 517–522 页。[32] C. G. Sathe、Y. Makris 和 B. C. Schafer，《MANTIS：删隐集成电路的机器学习近似建模》，载于《2023 年欧洲设计、自动化与测试会议暨展览会（DATE）》，IEEE，2023 年，第 1–6 页。[33] A. Rezaei、R. Afsharmazayejani 和 J. Maynard，《评估基于 eFPGA 的删隐算法的安全性》，载于《第 41 届 IEEE/ACM 计算机辅助设计国际会议论文集》，2022 年，第 1–7 页。[34] K. Shamsi、M. Li、D. Z. Pan 和 Y. Jin，《Cross-Lock：采用交叉开关架构的高密度布局级互连锁定》，载于《2018 年大湖区超大规模集成电路研讨会论文集》，2018 年，第 147–152 页。[35] J. Luu、J. Goeders、M. Wainberg、A. Somerville、T. Yu、K. Nasartschuk、M. Nasr、S. Wang、T. Liu、N. Ahmed 等，《VTR 7.0：面向 FPGA 的新一代架构与计算机辅助设计系统》，《ACM 可重构技术与系统汇刊（TRETS）》，第 7 卷，第 2 期，第 1–30 页，2014 年。[36] L. McMurchie 和 C. Ebeling，《PathFinder：一种基于协商的性能驱动 FPGA 路由器》，载于《1995 年 ACM 第三届现场可编程门阵列国际研讨会论文集》，1995 年，第 111–117 页。[37] J. Sweeney、M. J. Heule 和 L. Pileggi，《逻辑锁定建模技术》，载于《第 39 届计算机辅助设计国际会议论文集》，2020 年，第 1–9 页。[38] K. Shamsi、M. Li、D. Z. Pan 和 Y. Jin，《KC2：用于快速时序电路去混淆的密钥条件压缩》，载于《2019 年欧洲设计、自动化与测试会议暨展览会（DATE）》，IEEE，2019 年，第 534–539 页。