使用多FPGA系统进行功耗侧信道评估

Yaswanth Tavva 新加坡国立大学 新加坡
yaswanth@u.nus.edu

Geng Luo 新加坡国立大学 新加坡
luo.geng@u.nus.edu

Trevor E. Carlson 新加坡国立大学 新加坡
tcarlson@nus.edu.sg

Li-Shiuan Peh 新加坡国立大学 新加坡
peh@nus.edu.sg

arXiv:2611.32652v1 [cs.CR] 24 Nov 2026

摘要

复杂的乱序处理器在流片前难以评估功耗侧信道泄漏。单个现场可编程门阵列通常缺乏足够的容量，而传统的基于CAD的分析速度过慢，无法支持实际攻击所需的大量迹线。多FPGA 仿真能够提供更大的容量，但现有的分区方案缺乏安全感知能力，因此攻击性能较差。

本文提出了 Scissors，这是一种面向复杂乱序处理器流片前功耗侧信道评估的安全感知多FPGA分区方法。Scissors 使用侧信道脆弱因子（SVF）分析来识别与密钥相关的信号。随后，它将强密钥相关信号限制在一个 FPGA分区（测量分区）中，同时将其余信号移至其他 FPGA（支持分区），从而提高侧信道信息的可观测性。

我们通过寄存器传输级仿真以及在双 U200 FPGA测试平台上的物理测量来评估 Scissors，采用基于汉明重量泄漏模型的频域相关功耗分析。寄存器传输级消融实验表明，仅有 68–71 个高 SVF 信号主导密钥恢复过程。对于单核处理器（SingleBOOM），Scissors 仅需 12,000 条迹线即可恢复密钥，而未分区设计和 TritonPart （基线分区方法）分别需要 49,000 条和 43,000 条，同时将正确密钥相关性提高了 2.83×。对于四核处理器（QuadBOOM）而言，其容量超出了 CW340单FPGA平台的限制，Scissors 将所需迹线数量从 111,000 条降至 66,000 条，并将相关性在 TritonPart 基础上提高了 1.48×。Scissors 还将 QuadBOOM 从寄存器传输级到比特流的生成时间从 61 小时 39 分钟缩短至 6 小时 40 分钟，实现了 9.25× 的加速。这些结果表明，与基线方法相比，安全感知分区能够提高泄漏可观测性和攻击性能，同时显著缩短针对复杂处理器的实现时间。

1 引言

1.1 作为侧信道攻击目标的乱序处理器

物理侧信道研究传统上聚焦于密码加速器和顺序执行核心 [8, 37, 57]。然而，近期研究表明，复杂的乱序（OoO）处理器同样存在可利用泄漏 [10, 23]。随着乱序处理器被应用于高保证和边缘计算系统 [19]，理解这种泄漏变得愈发重要。

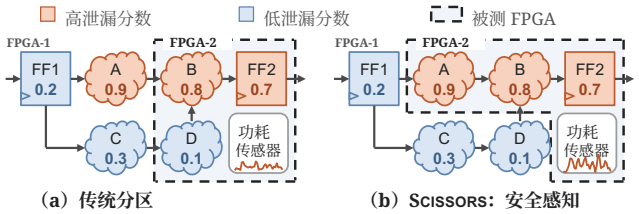


图 1：示例电路的多FPGA分区。(a) 传统分区会根据资源利用率、时序和 FPGA 间通信进行切分。(b) Scissors 根据 SVF分数进行切分，将高分模块保留在被测 FPGA 上。

乱序处理器涉及许多微架构事件，例如推测执行、动态调度、物理寄存器分配、转发和缓存。这些事件将依赖秘密的开关活动与大量无关的后台活动混杂在一起。因此，秘密计算产生的泄漏更难观测，要成功提取密钥就需要额外的迹线。例如，我们的结果表明，SiFive四核CPU最多需要10万条迹线才能提取未受保护的AES-128密钥（第6节）；而先前针对8级 [8] 和13级 [37] 顺序执行CPU的研究，只需不到1万条迹线便可提取密钥。

现有方法不支持针对大型乱序处理器的可扩展流片前泄漏评估。首先，软件级分析无法观测硬件事件，因此无法捕获秘密值在硬件中影响开关活动的位置。其次，基于 CAD 的仿真能够呈现开关活动，但对于大型乱序设计上的轨迹密集型攻击而言速度过慢，而此类攻击通常需要数千条 [10] 到数百万条 [36] 迹线。最后，FPGA 仿真提供了一种替代评估方案，能够实现更快的执行并收集物理测量结果，但通常无法将乱序处理器容纳在单个设备中。

1.2 多FPGA仿真与分区的作用

多FPGA仿真 [9, 15, 16, 25, 26, 32, 39, 43, 53, 60, 61]为超出单个设备容量的乱序处理器提供了一条可行的途径。最小割和 Tri- tonPart 等分区方法 [12] 通过应对 FPGA 容量限制实现了这一方案，但它们并非为侧信道测量而设计，并会引入意料之外的副作用。例如，它们会影响逻辑布局、将信号分布到不同器件上，并且

决定哪些信号和逻辑保留在一起 [55, 63]。因此，它们会无意中改变密钥相关活动与其他活动在测量轨迹中的组合方式。由此，功能等价的分区可能呈现不同的泄漏特性和攻击结果，导致需要更多测量轨迹，或只能恢复部分密钥字节，这表明有必要采用安全感知分区。如图 1(a) 所示，传统分区可能会将泄漏评分较高的模块 (A) 留在被测 FPGA 之外，而 Scissors 则会将此类模块保留在同一块板上（图 1(b)）。下一节将介绍总体概况。

1.3 方法概述

我们开发了 Scissors，这是一种面向复杂设计的多 FPGA 流片前功耗侧信道分析（SCA）评估框架。我们的框架利用 FPGA 仿真收集物理测量迹线，并支持超出单 FPGA 平台容量的设计。

我们将一块 FPGA 上测得的功耗轨迹分解为与安全相关的开关活动、无关开关活动和测量噪声：

$$P(t) = P_k(t) + P_o(t) + \eta(t), \tag{1}$$

其中，$P_k(t)$ 表示与密钥假设相关的活动，$P_o(t)$ 表示其他无关的微体系结构活动，$\eta(t)$ 表示测量噪声。相关功耗分析（CPA）试图区分 $P_k(t)$，而 $P_o(t)$ 和 $\eta(t)$ 会掩盖这种关系，并增加密钥恢复所需的迹线数量。

传统分区在不考虑这种泄漏构成的情况下，优化资源利用率、时序和 FPGA 间通信 [12, 27, 47]。因此，它可能会将与密钥相关的逻辑移出被测 FPGA，从而缩小 $P_k(t)$；或者将无关逻辑保留在其中，从而扩大 $P_o(t)$（图 1(a)）。这两种结果都会削弱泄漏可观测性，并增加密钥恢复开销。

为避免这种退化，Scissors 将侧信道信息直接纳入分区过程。它使用侧信道脆弱因子（SVF） [18] 分析，根据乱序核心信号与攻击假设的相关性对其进行排名。我们的结果表明，应用基线分区器 TritonPart [12], 可能会扰乱这些活动在测量中的呈现方式，因此需要与 Scissors 相比 3.58× 数量的迹线。由于 TritonPart 不具备安全感知能力，因此不知道这些信号应当放置在哪里。相反，Scissors 将依赖秘密的信号保留在被测 FPGA 的功耗域中，并将相关性较弱的活动移至辅助 FPGA，从而保留依赖密钥的活动的可观测性。因此，安全感知能力是必需的，因为功能等价的分区可能会暴露出截然不同的、由密钥相关切换和无关切换组成的混合信号。遵循最坏情况侧信道评估的实践，Scissors 优先保证泄漏可观测性，而不是允许偶然的无关活动被误认为安全性 [6]。需要指出的是，Scissors 不会消除物理测量噪声，也不会改变处理器功能。

图 5 表明，单独的无关信号无法实现密钥恢复，但当与和泄漏相关的信号一起测量时，会降低相关功耗分析性能。

我们在 XeonBOOM 和 QuadBOOM（表 3）上评估 Scissors，将其作为运行未受保护 AES 的代表性复杂乱序处理器。评估采用相关功耗分析（CPA）、密钥排名收敛、披露所需测量次数（MtD）以及实现开销，以量化迹线质量、攻击性能和仿真成本。总体而言，Scissors 将信号级泄漏估计转化为适用于需要多个 FPGA 的寄存器传输级设计的安全感知分区约束。因此，该方法同样适用于其他复杂处理器（包括数据中心 CPU），以及密码加速器等安全敏感RTL模块。

本文的贡献如下：

- 我们提出了 Scissors，这是一种由 SVF 引导、具备安全感知能力的多现场可编程门阵列分区方法，用于对复杂乱序处理器进行流片前功耗侧信道分析评估。Scissors 将泄漏相关逻辑映射到用于安全测量的现场可编程门阵列，并将其余逻辑分布到辅助 FPGA 上（图3）。
- 我们对 SingleBOOM、QuadBOOM 和 XeonBOOM 开展了寄存器传输级消融研究，结果表明，在 43,866–174,499 个信号中，仅有 68–71 个高 SVF 信号主导密钥恢复。在所有三种配置中，这些信号都能在 50 条迹线内恢复密钥；移除这些信号会增加 SingleBOOM 的 MtD 3.3×，并使 QuadBOOM 和 XeonBOOM 在迹线预算内无法恢复密钥（图 5 和表 4）。
- 我们使用双 U200 测试平台评估 Scissors 在 SingleBOOM 和 QuadBOOM 上的表现，并采用 HiFive P550 的测量结果作为实际硅芯片参考。对于 SingleBOOM，Scissors 需要 12,000 条迹线，而未分区设计需要 49,000 条，TritonPart 需要 43,000 条，使 MtD 最多降低 4.08×，并使正确密钥的相关性最多提高 2.83×。对于 QuadBOOM，Scissors 将 MtD 从 111,000 条迹线降至 66,000 条（1.68×），并将相关性提高 1.48×（图8和图9）。
- 我们实现了端到端侧信道分析工作流的自动化，并将其应用于多种乱序和顺序执行 CPU 配置（图10）。与 TritonPart 相比，Scissors 将 QuadBOOM 的 RTL到比特流时间从 61 小时 39 分钟缩短至 6 小时 40 分钟（9.25×），将 XeonBOOM 的预处理时间从 16 小时 48 分钟缩短至 2 小时 59 分钟（5.61×），同时将 Xeon-BOOM 分区接口从 18,946 比特缩减至 4,000 比特（表8和第7节）。

Scissors 也适用于密码加速器：对于 AES硬件实现，与未分区基线相比，它只需 4× 条更少的迹线即可恢复密钥。未分区实验使用一块 U200 FPGA，而 Scissors 使用由两块 U200 FPGA 组成的测试平台。总体而言，这项工作确立了分区作为可扩展流片前侧信道评估的关键手段，并为在硅片问世前分析复杂处理器设计提供了切实可行的途径。

概述。第2节介绍背景，第3节阐述多 FPGA 评估的动机。第4节介绍 Scissors 方法。第5节和第6节分别报告仿真和物理测量结果，第7节评估实现成本和可扩展性。第8节和第9节讨论相关工作与局限性，第10节总结全文。

2 背景

威胁模型。 我们考虑这样一种物理侧信道对手：它可以选择输入、触发重复的密码执行，并收集与功耗相关的迹线。我们在已知密钥设置下评估所有攻击，这是衡量泄漏强度、正确密钥排名和密钥恢复开销的常见做法。我们重点关注运行于乱序处理器上的未防护 AES-128 软件实现，并使用相关功耗分析（CPA）逐字节恢复密钥。

测量使用外部电压探针、电磁探针，或片上传感器 [3, 20, 44, 46, 54, 62]，通过捕获电压波动来表征功耗活动。我们的现场可编程门阵列设置使用布线延迟传感器（RDS） [48]，而在 HiFive P550 参考平台上的测量则使用 Keysight 示波器 [28]和 CT4204 Cal Test 弹簧加载探针 [13]，连接至 CPU 电源轨。

通过相关功耗分析利用泄漏。 相关功耗分析 [11] 将迹线与假设功耗模型进行相关。令 $\mathbf{T}_j = (T_d(j))_{d=1}^{D}$ 表示所有迹线中的采样点 j，并令 $\mathbf{H}_i = (HP_i(d))_{d=1}^{D}$ 表示密钥假设 K_i 对应的假设泄漏。我们将相关功耗分析的相关性写为 $R(i, j) = \mathrm{corr}(\mathbf{T}_j, \mathbf{H}_i)$，其中 $\mathrm{corr}(\cdot,\cdot)$ 表示皮尔逊相关。正确的假设应在 j 上产生最大的绝对相关性。时域相关功耗分析会考察迹线中的每个采样位置，但迹线之间的错位可能会降低相关性强度。

适配频域 [38, 40]。我们使用快速傅里叶变换（FFT）对迹线进行变换，以降低对迹线失配的敏感度。令 $\mathbf{F}_{j'} = (F_d(j'))_{d=1}^{D}$ 表示所有迹线在频率索引 j' 处的 FFT 幅值。我们沿用相同的相关性算子，并在 $N_{\mathrm{FFT}}/2$ 个频率分量上计算 $R(i, j') = \mathrm{corr}(\mathbf{F}_{j'}, \mathbf{H}_i)$。本文采用频域相关功耗分析。

安全性指标。我们使用得分向量 $\mathbf{p}$ 评估相关功耗分析（CPA），其中 p_i 表示密钥假设 K_i 在相关时间或频率分量上的最大绝对相关性。我们使用密钥排名和披露所需测量次数（MtD）作为指标。

密钥排名。令 $i^\star$ 表示正确密钥 $K_{i^\star}$ 对应的假设索引。密钥排名定义如下：

$$\mathrm{Rank}(K_{i^\star}) = 1 + \left|\{\, i \mid p_i > p_{i^\star} \,\}\right|. \tag{2}$$

披露所需测量次数（MtD）。MtD 是达到 $\mathrm{Rank}(K_{i^\star}) = 1$ 所需的最少迹线数。

侧信道脆弱因子（SVF） [18]。SVF 衡量内部信号活动与攻击假设之间的相关性。对于执行 d 中的信号值 $x_d[t]$ 和假设值 h_d，我们定义

$$a_d[t] = \mathrm{HD}(x_d[t], x_{d+1}[t]), \qquad o_d = \mathrm{HD}(h_d, h_{d+1}),$$
$$\mathrm{SVF}(x) = \max_{t \in \{1,\ldots,N_t\}} \left|\mathrm{corr}\big((o_d)_d, (a_d[t])_d\big)\right|. \tag{3}$$

其中，HD 表示汉明距离，SVF 分数复用了上文定义的皮尔逊相关算子。因此，较高的 SVF 分数表明信号的开关活动与攻击假设高度一致，从而更有可能促成成功的密钥恢复。

基于翻转计数的仿真 [56]。 PrimePower 和基于 SPICE 的模拟器能够提供高保真度的功率波形，但由于运行时间过长且许可成本高昂，难以扩展。因此，我们直接使用基于 VCD 信号状态的位归一化汉明重量活动轨迹作为功耗的代理。设 S 表示选定的设计信号，w_s 表示信号 s 的位宽。在时间采样点 t 处的归一化活动为

$$A_{\mathrm{HW}}^{\mathrm{norm}}[t] = \frac{\sum_{s \in \mathcal{S}} \mathrm{HW}\big(s[t]\big)}{\sum_{s \in \mathcal{S}} w_s}. \tag{4}$$

将所得迹线与每个密钥猜测对应的 AES S盒汉明重量假设进行相关性分析。表 1 列出了相关符号。有关各项指标的更详细讨论，请参阅 [42]。

表 1：术语与符号。

术语或符号	含义
功率与相关功耗分析符号	
$P(t)$	传感器在时间 t 处观测到的与功耗相关的迹线。
$P_k(t)$, $P_o(t)$, $\eta(t)$	密钥相关活动、其他微体系结构活动和测量噪声，分别为。
D	相关功耗分析方程中的迹线数量；在算法 1 中，为输入的 RTL 设计。
d	相关功耗分析和 SVF 中的迹线或执行索引；通信模型中的请求或响应阶段索引，其中 $d \in \{\mathrm{req}, \mathrm{resp}\}$。
$i, i^\star$	密钥假设索引和正确密钥假设索引，分别。
j, j'	时域采样索引和频域索引，分别。
t	功耗模型和 RTL 仿真中的时间/采样索引。
$T_d(j)$, $\mathbf{T}_j$	测量轨迹的第 j 个采样点 d，以及所有 D 条迹线中采样点 j 的向量，分别。
K_i、$K_{i^\star}$	由假设索引 i 表示的候选密钥和正确密钥，分别为。
$HP_i(d)$, $\mathbf{H}_i$	迹线 d 中密钥假设 K_i 对应的假设泄漏及其相应的分别为跨所有 D 迹线的向量。
$\mathrm{corr}(\cdot,\cdot)$	CPA 和 SVF 使用的皮尔逊相关运算符。
$R(i, j)$, $R(i, j')$	假设 i 在时域采样点 j 或频域索引 j'处的皮尔逊相关，分别为。
$F_d(j')$,$\mathbf{F}_{j'}$	迹线 d 在频率索引 j' 处的 FFT 幅值及其对应向量分别跨越所有 D 条迹线。
N_{FFT}	FFT 长度；实值迹线提供 $N_{\mathrm{FFT}}/2$ 个待分析频率分量-量。
$\mathbf{p}$, p_i	假设 i的 CPA 得分向量和最大绝对相关性在相关的时间或频率分量上。
排名($K_{i^\star}$)	根据 CPA 得分确定的正确密钥排名；第1名表示排名最高的候选项。
ρ	相关系数；$\lvert\rho\rvert$ 表示其绝对值。
SVF 与寄存器传输级仿真符号	
$x_d[t]$、h_d	RTL 信号在采样点 t处的值及相应的假设值在执行过程中 d。
$a_d[t]$、o_d	RTL 信号和泄漏预言机的汉明距离活动，分别为——别。
HD(·, ·), HW(·)	分别表示汉明距离和汉明重量运算符。
SVF(x), $\lvert\rho_{\mathrm{SVF}}\rvert$	分配给信号 x 的最大绝对相关性，以及等效的 SVF-用于信号选择的相关性幅值。
N_t	SVF 最大化过程中考虑的时间采样点数量。
$\mathcal{S}, s, w_s$	选定的信号集、该信号集中的一个信号及其位宽。
$x_{\mathrm{normHW}}[t]$	采样点 t处按位宽归一化的聚合汉明重量活动。
N	从 VCD 中提取的对齐工作负载窗口数量。
$\tau, \mathcal{S}_\tau$	SVF 阈值及满足该阈值的信号集合。
分区与通信符号	
$\mathcal{B}$、$\mathcal{X}$	分别表示黑盒后代和排除的实例集合。
I_r, I	去重后的选定 RTL 实例集合，以及该集合中的一个实例。
P_0、P_1	分别表示支持和被测 RTL 分区。 P_0 包含其余的支持逻辑，并在支撑 FPGA 上实现，而 P_1 包含所选实例，并在配备传感器的被测 FPGA 上实现。
K	多 FPGA 系统中的 FPGA 数量。
Γ	分区边界映射，包含跨界信号方向和位宽。
B_{uv}、B_{01}、B_{10}	从分区 u到 v的功能边界宽度，包括两个方向的双FPGA分区。
p、q_{uv}、F	每个数据包的功能位数、从 u 到 v的数据包数量，以及物理数据包宽度。
e, b	物理链路索引，以及通过该链路传输的物理位数。
$L_e(b)$, L_0, e, R有效 e	链路-e 的 b 比特延迟、其固定延迟及有效数据速率。多FPGA模型在按索引访问链路时使用等价形式 L_0，ℓ和 R有效 ℓ 按 ℓ索引链路。
T_{upd}	双FPGA请求-响应更新时间。
$U_{\max}$	一个逻辑内的跨分区稳定更新最大次数周期。
$T_{\text{仿真}}$, $f_{\text{仿真}}$	仿真时钟周期及其最大安全频率。
仿真_L, ℓ	物理链路集合及该集合中的一条链路。
r, $n(r)\,\ell$, d, b	请求-响应轮次索引、消息数量和物理比特数
$(r)\,\ell$, d	在第 r 轮的请求阶段或响应阶段 $d \in \{\mathrm{req}, \mathrm{resp}\}$ 期间，链路 ℓ 上。
$T(r)$	请求-响应轮次的持续时间 r。

3 现有硅前方法的局限性

现有的流片前侧信道分析方法无法同时提供足够的仿真速度、逼真的物理测量条件，以及复杂乱序（OoO）处理器所需的处理能力。门级仿真提供了详细的开关活动信息 [29], , 但其本质上没有噪声。此外，基于 BOOM 的乱序处理器的仿真速度每秒仅能运行几个处理器周期 [30], , 因此，收集实际攻击所需的大规模轨迹集会变得极其缓慢。SPICE 级噪声注入方法能够为小型设计提供更详细的物理模型 [5], , 但其计算开销使其难以应用于乱序处理器这类大型设计，因为运行时间过长。Scissors 仅使用寄存器传输级仿真来对信号进行排名，所需运行次数远少于相关功耗分析攻击所需的数万条迹线。

现场可编程门阵列仿真可显著提高执行吞吐量，并支持在实际运行条件下进行功耗测量。尽管 FPGA 查找表、布线资源和供电网络不同于 ASIC 标准单元和互连线，但二者都实现相同的寄存器传输级计算。因此，即使精确的泄漏幅度、时序和空间分布可能不同于 ASIC [50]，FPGA 测量仍可揭示这些计算是否会产生可通过统计方法利用的泄漏。ChipWhisperer [41], 、SAKURA [22],、FOBOS [21], 以及 OpenTitan CW310 [51] 和CW340 开发板 [52]等平台，为基于 FPGA 的侧信道分析提供了成熟的测量基础设施。每个平台都将目标设计部署在单个 FPGA 上。

然而，单个 FPGA 的容量限制了这种方法在复杂乱序设计中的应用。如图 2 所示，乱序配置（表 3）可能超出最大的现有侧信道分析板 CW340（表2）的容量。具体而言，配置 D9（表 3）与我们作为参考的四核 HiFive P550 最为接近，需要约 934k 个查找表，比 CW340 所支持的资源多 73%。增加发射宽度、重排序缓冲区大小、缓存容量或核心数量，可能会产生甚至超出 Alveo U200 [1]容量的设计。我们评估的 XeonBOOM 配置就是此类设计之一。多 FPGA 仿真通过将完整设计分布到多个设备上，解决了这一容量限制，
器件，同时保留现场可编程门阵列的执行速度和实际功耗测量能力。

表2：平台的资源可用性。

FPGA 平台	FPGA 器件	查找表 （查找表）(k)	寄存器 (k)	块 RAM (Mb)	DSP 切片	测量 Type
CW340 [2, 52]	XCKU095	537	1075	59	768	探针
Alveo U200 [1]	XCU200	1182	1956	67	5880	个片上传感器

4 Scissors：SVF 引导的安全感知分区

Scissors 使用 RTL 设计、代表性工作负载、泄漏模型、攻击假设以及目标 FPGA 平台。它在寄存器传输级对工作负载进行仿真，并根据侧信道脆弱因子（SVF）对信号进行排名（第2节）。这些评分指导图3所示的分区过程。Scissors 首先识别出包含强密钥相关信号的 RTL 实例，并将其

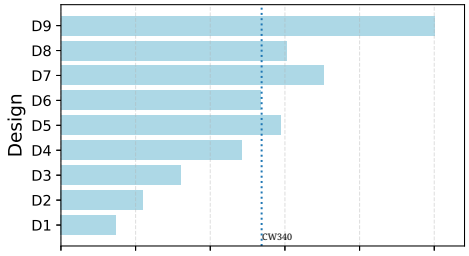


图 2：BOOM 各变体 D1–D9 的查找表使用情况。蓝色虚线表示 CW340/KU095 开发板的容量。

表 3：图 2 所示设计点的微架构配置。D10 [59], D11 [33], D12 [24] 是从文献中整理得出的调查点，并非现场可编程门阵列实现。

D1：Rocket64w1；D2：Rocket64x1；D3：SingleBOOM；D4：Rocket64z1；D5：Rocket64c9101；D6：Rocket64g1；D7：Rocket64gc401；D8：Rocket64x4；D9：QuadBOOM；D10：XeonBOOM；D11：类似 Lion Cove；D12：类似 Apple Firestorm。

参数	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
核心	1	1	1	1	1	1	1	4	4	1	1	1
解码宽度	1	2	3	4	4	5	6	2	3	6	8	8
ROB条目	32	64	96	128	192	130	216	64	96	510	576	~623
整数物理寄存器	52	80	100	128	160	128	115	80	100	280	~290	~380–394
浮点物理寄存器	48	64	96	128	160	128	132	64	96	332	~406	~432
加载队列条目	8	16	24	32	40	32	76	16	24	192	~189	~130
存储队列条目	8	16	24	32	40	32	45	16	24	114	120	~60
取指缓冲区条目	8	16	24	32	48	35	54	16	24	144	192†	–
L1-I	16 kB	16 kB	32 kB	32 kB	64 kB	32 kB	32 kB	16 kB	32 kB	32 kB	64 kB	192 kB
L1-D	16 kB	16 kB	32 kB	32 kB	64 kB	32 kB	32 kB	16 kB	32 kB	32 kB	48 kB	128 kB

固定到 被测 *FPGA* 上。随后，它将其余层次结构分配给 辅助 *FPGA*。这样，最有可能暴露秘密信息的逻辑便集中在被测 FPGA 上，而其余逻辑则被移至辅助 FPGA，从而提高与密钥相关的泄漏所占的相对贡献，即 $P_k(t)$。公式（1）刻画了这一效果。第5节解释了 Scissors 方法有效的原因，第6节则通过FPGA测量对此进行了展示。以下小节将详细介绍我们流程的各个阶段。

4.1 信号评分与 RTL 实例选择

测试平台将 RTL 活动和工作负载迭代标记记录在值变化转储（VCD）中。Scissors 将跟踪划分为N 与工作负载对齐的窗口，计算第 2 节中的 SVF 指标，并收集分数达到阈值 τ in S_τ 的信号。每次运行都会记录仿真层次结构、泄漏预言机、SVF 分数、SVF 阈值，以及与该阈值对应的信号集。

VCD 捕获仿真期间用于估计泄漏的信号活动。仿真允许我们有选择地将信号纳入分析，即聚焦于特定信号集，并排除其余信号。在实践中，这只能通过将信号实体地移动到不同 FPGA 上，并将其放置到我们选定的 FPGA 上来实现。因此，Scissors 会将 S_τ 中的每个信号映射到实现该信号的确切外层 RTL 实例。换言之，选择依据是实例的层级出现位置；

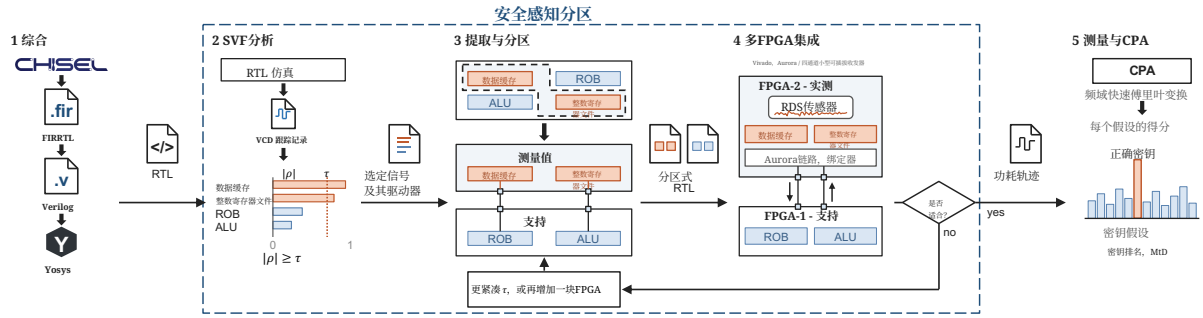


图3：Scissors 概览。

因此，选择一个实例并不会选择另一个核心中同一模块的其他实例。Scissors 会丢弃位于存储器宏、现有黑盒或其后代实例内部的候选项，因为这些实例的可综合 RTL 不可用。这些宏周围可见的逻辑仍符合条件，并且宏声明会为综合而保留。接下来，我们将详细介绍移动选定 RTL 实例的过程。

4.2 移动选定实例并验证每个边界

利用所得的实例集合，Scissors 将每个选定的层级出现位置恰好移动一次。以该出现位置为根的完整寄存器传输级层次结构将成为测量分区的一部分。如果选定的模块类型在其他位置被复用，Scissors 只会专门化从所选出现位置到设计顶层路径上的模块，而保持其他所有出现位置不变。

分离一个实例会将其现有接口暴露为分区边界。子模块输入会成为支持分区的输出和测量分区的输入，而子模块输出则沿相反方向传递。Scissors 通过专门化的祖先模块提升这些连接，并在边界映射 Γ 中记录每个信号、方向和位宽。第 4.1 步中选定的实例构成移动实例，用于形成测量分区 P_1；未改变的其余部分则形成支持分区 P_0。使用本地时钟。图 4 展示了这一过程。最后，临时顶层会直接重新连接 P_0 和 P_1；如果需要移动多个实例，Scissors 仅在原始应用级测试仍能完成时，才接受每个增量边界。算法 1 总结了这一构造过程。P_0 和 P_1 是同一设计的分区，分别在两块独立的 FPGA 上实现。这种方法可以推广到任意数量的支持分区，具体如第 4.4 节所述。下一节将介绍绑定器，它能够实现 FPGA 之间的无缝通信。

4.3 绑定器、资源闭包与频率

已接受的分区和边界映射 Γ 决定了两个 FPGA 为模拟一个同步设计必须交换哪些内容。对于每个方向，适配器会将跨分区信号打包成 B_{01} 位或 B_{10} 位向量。绑定器 会将这些向量封装成数据包，以适配有限位宽的物理链路 [7], 传输它们，并重新组装原始接收端的向量。数据包标签用于标识分区、逻辑周期、交换编号、向量位置和完成状态。

Alg算法 1SVF 引导的逐实例分区构建

输入：寄存器传输级 D、选定信号 S_τ、黑盒及其后代实例 B、排除的实例 X**输出：**支持 RTL P_0、测量 RTL P_1、边界映射 Γ1：丢弃 B 或 X 内的信号 2：$I_\tau \leftarrow$ 包含其余选定信号的唯一层次化 RTL 实例 3：**对于**每个选定实例 $I \in I_\tau$ **执行**4：沿层次路径从 I 到设计顶层克隆模块 5：分离完整实例 I，同时保留其 RTL 层次结构 6：将 I 的输入和输出连接提升为顶层端口 7：使用每个跨界信号的方向和位宽更新 Γ 8：在临时顶层中重新连接各分区；若验证失败，则拒绝该边界 9：**结束循环**10：将 I_τ 汇总到 P_1；将 P_0 设为剩余层次结构；输出 Γ 11：保持时钟局部；**返回** P_0，P_1，Γ

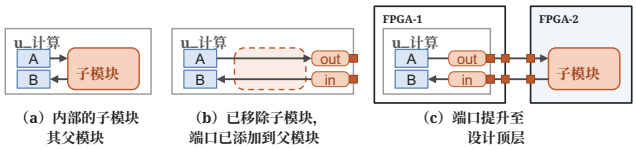


图 4：提取源逻辑。（a）所选信号的驱动单元被分组到其源实例内的一个子模块中。（b）移除该子模块会反转父模块中的每条连接：子模块输入变为父级输出。（c）端口逐级添加至设计顶层，设计顶层通过链路连接到被测 FPGA 上的子模块。

一次切分可能在一个逻辑周期内形成跨分区依赖链。P_0 将输入发送给 P_1，而由P_1 产生的输出又返回给 P_0。重新评估 P_0 后，可能会为 P_1产生另一个输入。由于每次交换都依赖于前一次交换的结果，绑定器会按顺序执行这些交换。这种行为在多*FPGA*分区中很常见，并且同时适用于*Scissors* 和 *TritonPart*。在最佳情况下，不会有组合逻辑

依赖跨越分区边界。此时，两个方向会并发传输，因此较慢的单向传输决定延迟。然而，跨越边界多次的组合路径可能需要最多 $U_{\max}$[1] 次更新。因此，我们选择 T_{emu} ，为所有交换完成以及所有跨越边界的值在下一个逻辑时钟沿之前稳定下来提供充足时间。 consequently，边界宽度和通信延迟决定了最高安全仿真频率。

Scissors 不使用通信成本来选择测量分区。SVF 首先进行此选择。随后，通信模型确定一个安全的时钟周期，以便其相关交换在一个逻辑周期内完成。如果每个数据包携带 p 功能位，则沿方向（u， v）传输边界需要 $q_{uv} = \lceil B_{uv}/p \rceil$ 个数据包。Scissors 对物理链路 e的特性描述如下：

$$L_e(b) = L_{0,e} + b/R_{\text{eff},e}, \tag{5}$$

其中， b 包括数据包头和填充，$L_{0,\ e}$ 是固定的链路延迟，而 $R_{\text{eff},\ e}$ 是有效数据速率。

考虑 40-Gb/s 的四通道小型可插拔收发器 FPGA 间链路 [4]。每个 256 比特的物理数据包携带 $p = 208$ 个功能位和 48 个元数据位。QuadBOOM Scissors 边界包含从 P_0 到 P_1 的 4,013 个比特，以及反向的 2,397 个比特（表8）。因此，绑定器发送 $q_{01} = \lceil 4,013/208 \rceil = 20$ 个数据包，以及 $q_{10} = \lceil 2,397/208 \rceil = 12$ 个数据包。计入数据包头和填充后，这些传输包含 $20 \times 256 = 5,120$ 和 $12 \times 256 = 3,072$ 个物理位。

使用物理数据包宽度 F 时，一次双 FPGA 更新需要

$$T_{\text{upd}} = L_{01}(q_{01}F) + L_{10}(q_{10}F). \tag{6}$$

由于返回向量取决于请求，全双工链路无法消除这一有序序列。逻辑仿真时钟必须满足

$$T_{\text{emu}} \geq U_{\max} T_{\text{upd}}, \qquad f_{\text{emu}} \leq 1/T_{\text{emu}}. \tag{7}$$

例如，让我们看看 QuadBOOM 实现的要求。将上述数据包数量代入公式（6）可得

$$T_{\text{upd}} = L_{01}(5,120) + L_{10}(3,072).$$

根据测得的 $L_{0,\ e} = 380\,\text{ns}$ 和 $R_{\text{eff},\ e} = 39.94\,\text{Gb/s}$，两个方向的传输分别需要 508.19 ns 和 456.92 ns，因此得到 $T_{\text{upd}} = 965.11\,\text{ns}$。我们的绑定器每个逻辑周期最多允许 $U_{\max} = 5$ 次相关更新，因此为 $T_{\text{emu}} \geq 4.83\,\mu\text{s}$ 和 $f_{\text{emu}} \leq 207.23\,\text{kHz}$。我们的实现保守地采用 100-kHz 处理器时钟，对应于 $T_{\text{emu}} = 10\,\mu\text{s}$。超过此周期的映射必须以更低的频率运行。

满足通信界限是必要条件，但并不充分，因为每个分区还必须容纳在其器件内。第 4.4 节接下来介绍如何扩展到两个以上的 FPGA，并且适用于 Scissors 和 TritonPart。

4.4 扩展至两个以上 FPGA

即使使用两块以上的 FPGA，Scissors 的安全决策仍保持不变：所有选定实例都集中在一块测量器件上。Scissors 只将支持逻辑分配到其余 $K-1$ 器件上，并遵循资源约束，即递归应用步骤 4.2 以创建新的分区。此外，增加辅助 FPGA 无法让过大的测量分区适配。这需要更改安全切分，例如提高 τ 并省略 SVF 较低的实例。同样，如果 k路 TritonPart 分区无法适配到 K 块 FPGA 上，我们会逐步增加分区和 FPGA 的数量。

按容量配置支持逻辑十分直接；主要挑战在于确定所有跨分区组合路径在下一个逻辑时钟沿到来前需要多长时间才能稳定。我们使用 T_{emu} 模型来选择足够长的逻辑时钟周期。在此期间，所有 FPGA 会反复交换当前的边界值，并评估其本地组合逻辑。

我们在将寄存器和存储器视为端点的同时，跟踪每条跨越分区边界的组合路径。任意此类路径跨越的 FPGA 边界的最大数量决定了一个逻辑时钟周期内所需的稳定轮次数。在每一轮中，FPGA 会交换当前的边界值，随后其本地组合逻辑会对新接收的值作出响应。逻辑时钟周期必须容纳足够多的轮次，使每条路径都能在下一个时钟沿到来前稳定于寄存器或存储器输入端。

如果链路 ℓ在请求-响应轮次 r 的阶段$d \in \{\text{req,resp}\}$中传输总计为 $b^{(r)}_{\ell,d}$比特的 $n^{(r)}_{\ell,d}$条消息，则该轮次的持续时间上界为

$$T^{(r)} = \sum_{d \in \{\text{req,resp}\}} \max_{\ell \in \mathcal{E}_L} \left(n^{(r)}_{\ell,d} L_{0,\ell} + \frac{b^{(r)}_{\ell,d}}{R_{\text{eff},\ell}} \right). \tag{8}$$

我们像在双现场可编程门阵列的情形中一样，将 $T^{(r)}$ 最多 $U_{\max}$ 个稳定轮次的结果相加，以确定安全的逻辑时钟周期。应用级仿真和物理时序收敛则提供最终验收测试。

4.5 时序感知基线

为单独评估安全感知切分的优势，我们将其与 TritonPart 进行比较。TritonPart 是 OpenROAD 的时序感知超图划分器。TritonPart 对综合单元和共享网络超边进行分区，以满足时序和平衡约束；它不对 SVF 分数或 RTL 实例路径建模。两种方法均从相同的 RTL 出发，并采用相同的黑盒策略、现场可编程门阵列数量和平台约束。两者的区别在于可移动单元和优化目标：TritonPart 切分单元以优化物理成本，而 Scissors 则移动因其与密钥相关的活动而被选中的精确 RTL 实例，如第 4.1 节和第 4.2 节所述。

第 4.3 节中的绑定器、通信模型、资源检查和功能验证，以及第 4.4 节中的缩放方法，共同构成 TritonPart 和 Scissors 的通用实现层。我们对两种方法采用完全相同的实现层，以确保比较能够单独评估边界选择策略。

4.6 执行流程

在两个分区均通过功能检查和可行性检查后，我们使用 Vivado 生成它们的比特流。随后，我们对 FPGA 进行编程、初始化链路，并校准被测 FPGA 上的片上传感器

[1]我们通过静态跟踪跨越分区边界的组合逻辑依赖链，$U_{\max}$ 在分区后立即确定

。采集控制器将传感器的 100 MHz 采样时钟与仿真设计时钟关联起来，确定块随机存取存储器记录窗口的大小，并使用软件标记界定 AES-128 S 盒的执行范围。在一次加密过程中，绑定器为每个逻辑周期执行所需的边界更新，同时传感器记录细粒度的传播延迟变化，作为功耗的代理指标。在标记区域结束时，被测 FPGA 将缓冲的迹线流式传输至主机并报告完成；随后，软件即可开始下一次加密。这些迹线之后将用于发起相关功耗分析攻击。

5 仿真评估

本节使用 RTL 仿真来确定高 SVF 信号对基于相关功耗分析的密钥恢复有多大影响。RTL 仿真能够查看各个信号的取值，让我们可以选择信号组，并测量它们对密钥排名和披露所需测量次数（MtD）的影响。我们首先单独评估高 SVF 信号，然后将它们与所有选定信号组合，最后移除这些信号，以量化它们在 Xeon-BOOM、QuadBOOM 和 SingleBOOM 中的贡献。

在本实验中，每种处理器配置均执行 AES 的未加保护软件实现。我们使用汉明重量泄漏模型进行相关功耗分析，并根据密钥候选项与仿真 RTL 活动的相关性对其进行排名。

为进行消融研究，我们根据表 4 所列的 SVF 相关性范围对信号进行划分。尽管每种配置都包含数万个选定信号，但其中不足 100 个信号具有 $|\rho_{\text{SVF}}| \geq 0.9$。图 5 显示，这一小部分高 SVF 信号在密钥恢复中起主导作用。

我们首先仅使用具有 $|\rho_{\text{SVF}}| \geq 0.9$ 的信号执行相关功耗分析。对于三种配置，相关功耗分析均能在 50 条迹线内恢复正确密钥，表明这些信号包含高度集中的密钥相关泄漏。接下来，我们纳入所有选定信号，以考察较弱信号对攻击的影响。对于 SingleBOOM 和 QuadBOOM，相关功耗分析仍能恢复密钥，但与仅使用高 SVF 信号的实验相比，分别需要约 6× 和 16× 条额外迹线。对于 XeonBOOM，虽然正确密钥仍位列前三候选，但未能将其提取出来。因此，将高 SVF 信号与数量多得多的较弱信号结合起来，虽然保留了有用的泄漏，但降低了攻击性能。

为单独评估高 SVF 子集的重要性，我们随后排除了所有具有 $|\rho_{\text{SVF}}| \geq 0.9$ 的信号。尽管每种配置仅移除了 68–71 个信号，这一删减仍产生了显著影响。SingleBOOM 需要近 700 条额外迹线才能恢复密钥，相当于增加了 3.3× 。更重要的是，在迹线预算内，相关功耗分析不再能为 QuadBOOM 或 XeonBOOM 恢复密钥。

进一步缩小信号集会强化这一趋势。当我们仅保留具有 $|\rho_{\text{SVF}}| < 0.5$ 的信号时，SingleBOOM 的正确密钥排名显著恶化，而 QuadBOOM 和 XeonBOOM 仍能抵御密钥恢复。因此，结果表明，低 SVF 信号无法弥补小规模高 SVF 子集被移除所造成的影响。

总体而言，高 SVF 信号提供了大部分泄漏，使基于相关功耗分析的密钥恢复成为可能。单独分析这些信号时，它们能够高效地揭示密钥；即使与较弱的信号结合，它们仍能继续支持密钥恢复

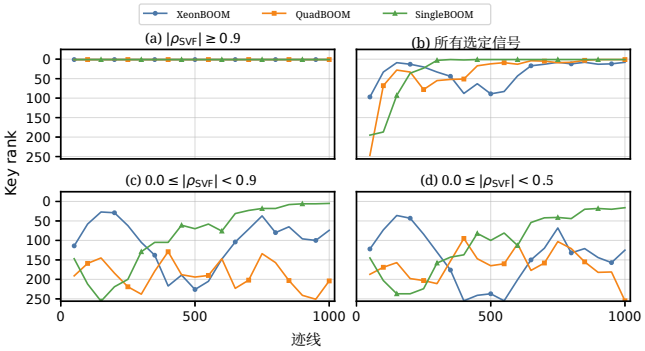


图 5：针对 Xeon-BOOM、QuadBOOM 和 SingleBOOM CPU 配置中基于 SVF 的信号组发起的相关功耗分析攻击。纵轴排名越低越好。仅使用高 SVF 信号（子图 (a)）时，三种设计均可在 50 条迹线内恢复正确密钥，这在仿真中确立了一个下界。移除这些信号（子图 (c) 和 (d)）会阻止 QuadBOOM 和 XeonBOOM 的密钥恢复，并使 SingleBOOM 的表现恶化，凸显了这些信号的重要性。子图 (b) 同时包含与秘密密钥相关的信号活动和背景信号活动，因此需要更多迹线，这表明无关活动会增加攻击所需的工作量。

表 4：图 5 中每个 SVF-相关性范围内选定信号的数量。

SVF 选择	XeonBOOM	QuadBOOM	SingleBOOM
$\lvert\rho_{\text{SVF}}\rvert \geq 0.9$	71	68	68
所有选定信号	174,499	66,180	43,866
$0.0 \leq \lvert\rho_{\text{SVF}}\rvert < 0.9$	174,428	66,112	43,798
$0.0 \leq \lvert\rho_{\text{SVF}}\rvert < 0.5$	174,374	66,097	43,776
$0.0 \leq \lvert\rho_{\text{SVF}}\rvert < 0.1$	173,978	65,509	43,452

，尽管较弱的信号会增加 MtD。相比之下，移除高 SVF 信号会使两种配置无法进行密钥恢复，并使第三种配置中的恢复效果下降。

这项寄存器传输级研究涵盖 TileGroup，其中包括处理器核心和 L1 缓存。下一节评估的现场可编程门阵列实现涵盖完整的片上系统，包括 L2 缓存、DDR 控制器和外围子系统。这些发现为我们的安全感知分区策略提供了指导：将高 SVF 信号的放置限制在特定的现场可编程门阵列上，同时将低 SVF 活动移至辅助 FPGA。

6 测量评估

本节评估在寄存器传输级仿真中识别出的依赖秘密的活动，在现场可编程门阵列实现后是否仍然可观测。我们首先使用 HiFive P550 平台建立 ASIC 参考，然后利用双 U200 FPGA 测试平台评估 SingleBOOM 和 QuadBOOM 上的 Scissors，最后将研究扩展到

[2]AES-128 参考设计来自密码硬件 IP 代码库：https://github.com/ECSIS-lab/Cryptographic-Hardware-IPs。

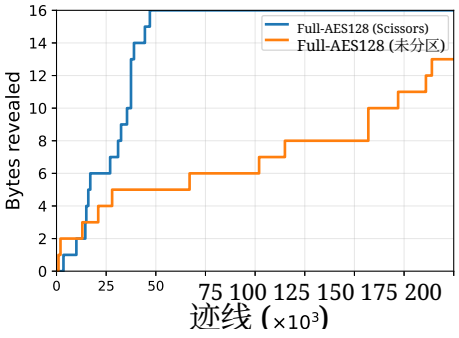
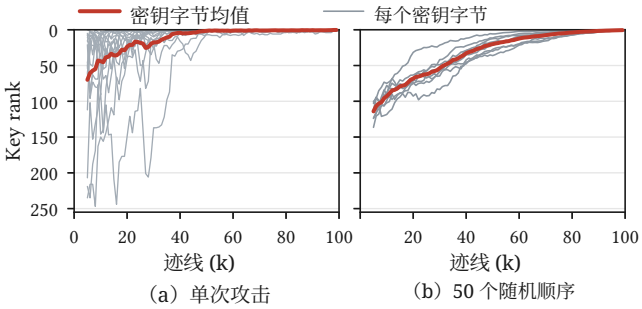


图 6：Full-AES128 的字节揭示进度。我们比较了相同的 Full-AES128 设计[2] 在分区式（Scissors）和未分区形式下进行相关功耗分析的结果。我们将某个字节在其首次出现排名第1的迹线时视为已揭示；如果从未出现排名第1的迹线，则在其首次出现排名第2的迹线时视为已揭示。Scissors 在大约 47,000 条迹线时揭示全部 16 个字节，而未分区形式在 200,000 条迹线以内未能揭示所有字节。



（a）单次攻击　（b）50 个随机顺序

图 7：在 HiFive P550 上进行相关功耗分析。（a）分别对 16 个密钥字节进行单次攻击。（b）对密钥字节 8 至 15 分别进行 50 次攻击并取平均值，每次攻击均使用 100,000 条迹线的随机排列。灰色线表示各个密钥字节，红线表示其平均值。

表5：针对相同攻击假设的 SingleBOOM 配置相关功耗分析结果比较。

攻击结果				Scissors 改进		
设计	U200	迹线	$\lvert\rho\rvert$	基线	迹线缩减	$\lvert\rho\rvert$ 增益
未分区	1	49k	0.01240			
TritonPart	2	43k	0.01144	未分区	**4.08×**	**2.61×**
Scissors	2	**12k**	**0.03236**	TritonPart	**3.58×**	**2.83×**

硬件 AES 和 PRESENT 加速器。如表 3 所示，QuadBOOM 的配置与 P550 配置高度相似 [34, 35]。此外，QuadBOOM 的容量超过了 CW340——目前可用的最大单 FPGA 侧信道平台——因此是多 FPGA 评估的代表性候选平台。在这些平台上，我们在相同的汉明重量泄漏模型下比较披露所需测量次数（MtD）和秘密密钥相关性。

HiFive P550 黄金参考。 我们从运行于 Ubuntu 24.04 的 HiFive P550 上未受保护的 AES-128 收集了 100,000 条功耗轨迹。GPIO 触发信号用于标识 S 盒窗口，并且

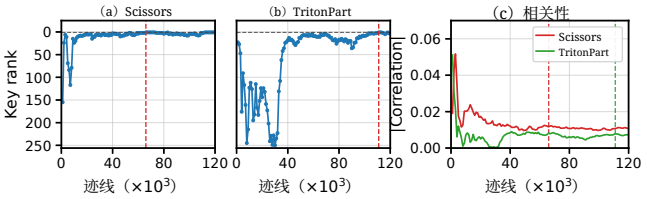


图8：QuadBOOM 中 Scissors 与 TritonPart 的相关功耗分析攻击对比。子图 (a) 和 (b) 分别展示 Scissors 和 TritonPart 在频域相关功耗分析中的密钥排名变化，子图 (c) 则比较二者的正确密钥绝对相关性。排名越低越好；相关性越高越好。QuadBOOM 的四核逻辑使 Scissors 的泄露成本从 SingleBOOM 的 12,000 条迹线提高到 66,000 条迹线。由于 Scissors 集中依赖秘密的信号，同时排除了无关逻辑，因此该结果仍构成 FPGA 实现中的低噪声界限。TritonPart 未考虑安全性，反而无意中引入了无关切换噪声，因此需要 111,000 条迹线。Scissors 实现了 1.48× 更高的秘密密钥相关性。

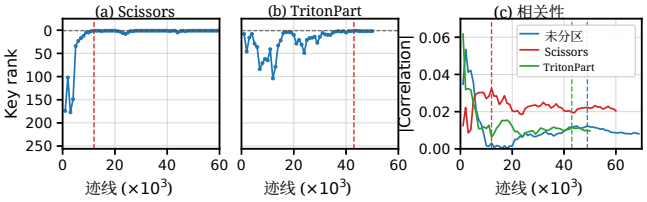


图9：SingleBOOM 中 Scissors 与 TritonPart 的相关功耗分析攻击对比。子图 (a) 和 (b) 分别展示 Scissors 和 TritonPart 在频域相关功耗分析中的密钥排名变化，子图 (c) 则比较二者的正确密钥绝对相关性。密钥排名越低越好；相关性越高越好。分区式设计使用两块 U200。Scissors 使用 12,000 条迹线即可披露密钥，而未分区设计需要 49,000 条，TritonPart 需要 43,000 条；相关性分别提高了 2.61× 和 2.83×。通过将依赖秘密的逻辑集中在测量分区中，Scissors 为 FPGA 实现提供了低噪声界限。

表6：分区部署在两块 U200 FPGA 上的 QuadBOOM 的攻击性能。迹线表示正确密钥达到第1名时的首个迹线数量，$\lvert\rho\rvert$ 是相应的绝对相关值。

攻击结果				Scissors 改进		
设计	U200	迹线	$\lvert\rho\rvert$	基线	迹线缩减	$\lvert\rho\rvert$ 增益
TritonPart	2	111k	0.00788	TritonPart	**1.68×**	**1.48×**
Scissors	2	**66k**	**0.01167**			

是德科技示波器以 1 GS/s 的采样率对 0.8 V 的 CPU 电源轨进行采样。由于拆除了散热器以便接入电源，处理器以 500 MHz 的频率运行，而非其 1.4 GHz 的峰值频率。

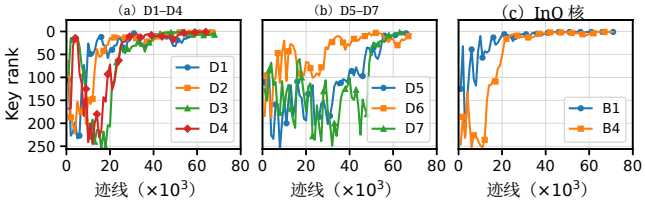


图10：不同处理器配置下的相关功耗分析攻击性能。表 3 定义了 D1–D7。子图 (a) 考察 D1–D4，这些设计可以部署在 CW340 上；CW340 是可用的最大侧信道板卡。子图 (b) 考察 D5–D7，这些设计超出了 CW340 的容量；子图 (c) 则比较了 B1 顺序单核设计和 B4 四核设计。我们的自动化流程能够评估这一大范围的设计。排名越低越好。更简单的 B1 和 B4 设计在大约 40,000 条迹线后达到第1名（恢复密钥）。D1–D4 需要约 60,000 条迹线，而 D5–D7 需要约 70,000 条迹线。

表 7：Scissors 和 TritonPart 为 AES 和 PRESENT 生成的分区边界大小比较。TritonPart 为 AES 生成 520 比特的边界，为 PRESENT 生成 307 比特的边界。

	AES		PRESENT	
SVF 阈值	边界 Bits	小于 TritonPart	边界 Bits	小于 TritonPart
≥ 0.9	256	2.03×	256	1.20×

图7(a)显示，大多数密钥字节在约 60,000 条迹线内达到第1名，全部 16 个字节则在 100,000 条迹线内达到第1名。为消除迹线顺序造成的偏差，我们还针对每个字节 8–15 执行了 50 次随机化相关功耗分析攻击。在这些攻击中，40,000、60,000 和 80,000 条迹线时的平均正确密钥排名分别为 34、13 和 5（图 7(b)）。完整密钥恢复约需 100,000 条迹线。成功恢复密钥证实，在真实的电源轨和电路板噪声环境下，我们在整个评估过程中采用的 AES S盒汉明重量假设能够暴露商用乱序处理器上的可利用泄漏。后续的 FPGA实验会在硅片之前评估相同的泄漏假设，而通过比较未分区、TritonPart 和 Scissors 实现，可以在使用相同寄存器传输级和测试平台的情况下，隔离分区的影响。对于后续所有使用 FPGA测量的 CPU评估，我们仅展示 1 字节攻击。

SingleBOOM。 图9和表5对比了SingleBOOM的未分区、TritonPart和Scissors实现。Scissors仅使用 4.08× 更少的迹线即可达到第1名，比TritonPart少使用 3.58× 条迹线。我们还观察到，Scissors实现了更高的秘密密钥相关性：分别比未分区设计和TritonPart高 2.61× 和 2.83× 。在无噪声仿真中，正确密钥仅用300条迹线即可达到第1名。由于各种物理噪声源和无关切换，物理测量需要更多迹线。测量结果表明，与未分区设计和TritonPart相比，Scissors更好地保留了所识别的泄漏。

QuadBOOM。QuadBOOM无法容纳于CW340中。CW340是我们设置中可用的最大单FPGA侧信道平台，因此多FPGA分区不可避免。在我们的评估中，TritonPart产生了对称划分：核心0和核心3及其私有L1缓存被映射到测量分区，而核心1和核心2及其L1缓存被映射到支持分区。然而，TritonPart并非安全感知型工具，也不会优化哪些泄漏相关逻辑应放置在被测 FPGA中。相比之下，Scissors会明确地将此类逻辑映射到测量分区。

如表6所示，Scissors 需要 66,000 条迹线才能达到第1名，而 TritonPart 需要 111,000 条迹线，使 MtD 降低了 1.68×。我们还观察到，Scissors 的正确密钥相关性提高了 1.48× 。作为参考，无噪声的 QuadBOOM寄存器传输级仿真只需 800 条迹线即可达到第1名，而在物理 HiFive P550 上进行完整密钥恢复则大约需要 100,000 条迹线。Scissors 较低的迹线需求代表了一个经验下界，因为它保留了相同的寄存器传输级设计，同时将大量无关切换排除在测量分区之外。总体而言，这些结果展示了理想化仿真与物理测量之间的差距，而 QuadBOOM 的对比表明，安全感知分区所需的迹线数量少于基线分区方法。

超越中央处理器。我们还将 Scissors 应用于专用 AES-128 和 PRESENT 加速器。对于 AES，Scissors 将所需迹线数量减少了超过 4×，并恢复了全部 16 个密钥字节，而未分区设计只能恢复 13 个。我们的方法不会将传感器位置限制在高 SVF 逻辑附近，因此我们遍历不同的传感器布局，以验证密钥恢复不依赖于有利位置。在四个位置重复攻击，包括默认传感器位置 [48] 以及 U200 三个超级逻辑区域（SLR0、SLR1 和 SLR2）各自的一个位置后，我们在每个位置都恢复出了密钥，表明结果并不依赖于某个有利布局。我们采用 AES 加速器，是因为每种布局的构建时间约为 5 分钟，这使其相比 QuadBOOM 更适合进行类似的遍历（表8）。对于 AES 和 PRESENT，Scissors 的表现也都优于 TritonPart（表7），证明了该方法能够推广到处理器工作负载之外。

在 P550、SingleBOOM、QuadBOOM 和硬件加速器上进行的测量结果证实，RTL 可见泄漏在物理实现后依然存在。更重要的是，与基线分区方法相比，Scissors 始终能提升排名收敛速度和正确密钥相关性。

7 实现成本与可扩展性 lity

本节评估实现上述测量改进所需的成本。我们考察 RTL 到比特流时间、SVF 分析与划分时间、分区接口宽度、FPGA 利用率，以及扩展到更大型乱序核心的可扩展性。

分区生成成本。TritonPart 需要综合网表才能进行图划分。在我们的流程中，Yosys 首先将设计映射到 ASIC 标准单元库，随后 TritonPart 对生成的门级网表进行分区。尽管这些分区保留了所需的功能，但它们针对的是 ASIC 技术，而非 FPGA。因此，Vivado 必须重新综合每个分区，并将其重新映射到 U200 资源。在 FPGA 评估中使用 TritonPart 的输出时，这一额外映射步骤不可避免

表8：QuadBOOM 的实现成本。

分区	接口位 F1→F2 / F2→F1（总计）	FPGA 1 CLB 查找表	FPGA 2 CLB 查找表	构建时间（时:分）
TritonPart	273/487（760）	694,707（58.76%）	620,610（52.49%）	61:39
Scissors	4,013 /2,397（6,410）	919,795（77.80%）	75,593（6.39%）	**6:40**

。相比之下，Scissors 会执行工作负载引导的 SVF分析，并在分区之间移动高 SVF 的 RTL 实例。生成的分区保留其 RTL 表示形式，可直接进入 FPGA综合流程。QuadBOOM 的 SVF分析耗时 318.89 秒，XeonBOOM 耗时 1,506.39 秒。对于 XeonBOOM，完整的 Scissors 预处理流程耗时 2 小时 59 分钟，而 Yosys–TritonPart流程耗时 16 小时 48 分钟，提速 5.61× 倍。

现场可编程门阵列实现成本。 表8报告了QuadBOOM 实现的逻辑分区接口、最终布局后的CLB LUT利用率，以及完整的RTL到比特流时间。

保留与泄漏相关的寄存器传输级代码的代价是逻辑接口更宽：Scissors 使用 6,410 个接口位，而 TritonPart 使用 760 个。TritonPart 将两个核心及其 L1 缓存放置在 FPGA 2 上，因此 FPGA 2 上使用了 620,610 个 CLB 查找表，FPGA 1 上使用了 694,707 个。Scissors 则将大部分处理器留在 FPGA 1 上，仅将选定的五个高 SVF 实例移至 FPGA 2。因此，FPGA 2 仅使用 75,593 个 CLB 查找表，而 FPGA 1 使用 919,795 个。

TritonPart 所需的 ASIC 网表重映射也增加了 Vivado 构建时间。在整个 FPGA流程中保留寄存器传输级代码后，Scissors 将完整的 RTL到比特流生成时间从 61 小时 39 分钟缩短至 6 小时 40 分钟，提速 9.25×。Scissors 还只将与泄漏相关的实例放置在被测 FPGA 上，大幅降低了 FPGA 2 的利用率，尽管由此产生的分区接口更宽。在我们的实现中，这种更宽的接口使 Scissors 的整体仿真速度变慢。然而，Scissors 大幅减少了所需的迹线数量[3]，从而使整体攻击速度更快。对于 QuadBOOM 的单字节攻击，Scissors 耗时 2.35 小时，而 TritonPart 耗时 3.95 小时，因此 Scissors 快 1.68×。我们预计攻击时间会随密钥字节数成比例增长。因此，使用 Scissors 进行 16 字节攻击约需 37.56 小时，使用 TritonPart 则约需 63.16 小时，实际攻击时间缩短约 25.60 小时。

8 相关工作

针对复杂处理器的侧信道攻击。先前研究表明，现代处理器会通过基于遥测和基于探针的测量泄漏信息。Hertzbleed [58] 利用频率缩放效应，将功耗泄漏转化为时序泄漏。PLATY-PUS [36]和 Collide+Power [31]利用软件可访问的功耗遥测来观测处理器级别的泄漏。其他研究则对真实处理器进行物理功耗或电磁测量，包括 Apple vs. EMA [23], 、SpectrEM [17], 和 Reality Check on Side-Channels [10]。这些研究证实，复杂处理器在实际执行期间可能泄漏机密信息，也因此推动了流片前框架的发展，以便在制造前评估此类泄漏，并将规模扩展到单 FPGA 仿真所能达到的范围之外。

[3]每条迹线包含 60 个采样点，每个采样点为 128 比特，总计 960 字节。按照被测 UART 吞吐量 7.5 kB/s 计算，传输一条迹线需要 0.128 秒。

基于 FPGA 的处理器研究提供了一种流片前替代方案。Desmet 等人 [19] 证明，乱序执行可能破坏软件掩蔽背后的假设，但他们的研究采用的是简化的乱序 RISC-V 处理器。CryptRISC [49] 研究了一种具备安全加密执行硬件支持的顺序 RISC-V 处理器。现代乱序设计拥有更大的重排序缓冲区、物理寄存器文件、发射队列和缓存，这些都会增加泄漏的复杂性和 FPGA 资源需求 [14]。Scissors 针对这一尚未充分探索的场景，通过支持采用多 FPGA 仿真的更大型乱序设计，并将分区视为一种能够提升泄漏可观测性的因素。

多 FPGA 仿真与分区技术。 以往的多FPGA分区工作主要优化容量、时序和通信。TritonPart 针对物理设计执行约束驱动的图分区 [12]。FireAxe 使用用户引导的模块提取，将大型 BOOM 设计分区到多个 FPGA 上 [59]。更早期的处理器原型（例如 Intel Nehalem FPGA 仿真）采用微体系结构聚类来管理 FPGA 间通信 [45]。这些方法提高了仿真的可扩展性，但在分区过程中没有考虑侧信道泄漏。

Scissors 将泄漏引导机制引入多FPGA分区。它使用 SVF 分数，将密钥相关信号放置在被测 FPGA分区中，并评估侧信道收益和实现成本。

9 讨论

通用性与适用性。 Scissors 通过在 FPGA 分区之间重新分配逻辑，同时考虑安全数据的泄漏，保留了原始的寄存器传输级行为。给定工作负载、泄漏假设和仿真活动，该方法不仅适用于 AES 和乱序核心，也适用于其他硬件。此外，通过利用片上传感器 [48], ，该框架无需外部示波器。借助自动化，我们使用物理测量迹线评估了广泛的 CPU 配置（见图10）。我们计划将该框架开源，以支持对其他工作负载和泄漏假设的进一步评估。

局限性。 该方法支持流片前评估，但不能替代最终的硅片测量。FPGA 查找表、布线和供电与 ASIC 实现不同，因此绝对迹线数量无法直接迁移。能够捕获真实测量噪声的快速 ASIC 仿真仍是一个开放问题。

10 结论

本文介绍了 Scissors，这是一种由 SVF 引导的多FPGA分区流程，用于对复杂乱序处理器进行流片前功耗侧信道评估。Scissors 能识别密钥相关信号、提取相应逻辑，并将其映射到被测 FPGA，以提升 CPA 期间的泄漏可观测性。结果表明，分区不仅是将大型设计部署到多个 FPGA 上的一种实现手段，也是一项与安全相关的设计选择，会影响迹线质量和攻击性能。在各项物理 FPGA 评估中，与相应基线相比，Scissors 将 MtD 最多降低 4.08×，将正确密钥相关性最多提高 2.83×，并将 Quad-BOOM 的 RTL 到比特流生成时间缩短 9.25× 。

参考文献

[1] Advanced Micro Devices, Inc. [不详]。AMD Alveo U200 数据中心加速卡。https://www.amd.com/en/products/accelerators/alveo/u200/a-u200-a64g-pq-g.html。访问日期：2026-09-16。[2] Advanced Micro Devices, Inc. 2016。UltraScale FPGA 产品表和产品选择指南（XMP102）。https://docs.amd.com/v/u/en-US/ultrascale-fpga-product-selection-guide。文档 ID：XMP102；版本：1.7。访问日期：2026-09-13。[3] Sohrab Aftabjahani、Ryan Kastner、Mark Tehranipoor、Farimah Farahmandi、Jason Oberg、Anders Nordstrom、Nicole Fern 和 Alric Althoff。2021。专题环节：硬件安全计算机辅助设计——自动化是解决方案得以采用的关键。载于 2021 年 IEEE 第 39 届超大规模集成电路测试研讨会（VTS）。IEEE，1–10。[4] AMD Xilinx。2022。在 Alveo 中通过 Vitis 流程使用 Aurora IP。https://xilinx.github.io/Vitis-Tutorials/2022-1/build/html/docs/Hardware_Acceleration/Design_Tutorials/08-alveo_aurora_kernel/README.html。Vitis 教程 2022.1，访问日期：2026 年 10 月 4 日。[5] Md Toufiq Hasan Anik、Jean-Luc Danger、Sylvain Guilley 和 Naghmeh Karimi。2024。探讨受保护的掩码替换盒在温度和老化失配条件下面对模板攻击时的韧性。IEEE 超大规模集成电路系统汇刊 32，5（2024），911–924。[6] Melissa Azouaoui、Davide Bellizia、Ileana Buhan、Nicolas Debande、Sébastien Duval、Christophe Giraud、Éliane Jaulmes、François Koeune、Elisabeth Oswald、François-Xavier Standaert 等。2020。对侧信道评估策略的系统性评估。载于安全标准化研究国际会议。Springer，46–66。[7] Jonathan Babb、Russell Tessier 和 Anant Agarwal。1993。虚拟连线：克服基于 FPGA 的逻辑仿真器中的引脚限制。载于 *[1993]* IEEE 定制计算机用 FPGA 研讨会论文集。IEEE，142–151。[8] Josep Balasch、Benedikt Gierlichs、Oscar Reparaz 和 Ingrid Verbauwhede。2015。1 GHz 下的差分功耗分析、位切片与掩码。载于密码硬件与嵌入式系统国际研讨会。Springer，599–619。[9] Christophe Bobda、Joel Mandebi Mbongue、Paul Chow、Mohammad Ewais、Naif Tarafdar、Juan Camilo Vega、Ken Eguro、Dirk Koch、Suranga Handagala、Miriam Leeser 等。2022。数据中心与云中的 FPGA 加速未来。ACM 可重构技术与系统汇刊（TRETS）15，3（2022），1–42。[10] Harishma Boyapally、Dirmanto Jap、Qianmei Wu、Fan Zhang 和 Shivam Bhasin。2025。Reality Check on Side-Channels：从在采用乱序执行的 ARM Cortex-A72 处理器上破解 AES 中汲取的经验。载于第 19 届 USENIX 攻击性技术研讨会（WOOT 25）。179–189。[11] Eric Brier、Christophe Clavier 和 Francis Olivier。2004。基于泄漏模型的相关功耗分析。载于密码硬件与嵌入式系统国际研讨会。Springer，16–29。[12] Ismail Bustany、Grigor Gasparyan、Andrew B Kahng、Ioannis Koutis、Bodhisatta Pramanik 和 Zhiang Wang。2023。一种面向超大规模集成电路物理设计的开源约束驱动通用分区多工具。载于 2023 年 IEEE/ACM 国际计算机辅助设计会议（ICCAD）。IEEE，1–9。[13] Cal Test Electronics。2018。CT4200 系列数据表：无源电压示波器探头。https://as.caltestelectronics.com/public/attachments/CT4200_Series_datasheet.pdf。访问日期：2026-10-02。[14] Yun Chen、Ali Hajiabadi、Romain Poussier、Yaswanth Tavva、Andreas Diavastos、Shivam Bhasin 和 Trevor E Carlson。2025。PARADISE：面向功耗攻击防护的关键性知情指令重排。ACM 架构与代码优化汇刊 22，1（2025），1–25。[15] Grigory Chirkov 和 David Wentzlaff。2023。Smappic：云端可扩展多 FPGA 架构原型平台。载于第 28 届 ACM 编程语言与操作系统架构支持国际会议论文集，第 2 卷。733–746。[16] Guohao Dai、Tianhao Huang、Yuze Chi、Ningyi Xu、Yu Wang 和 Huazhong Yang。2017。ForeGraph：探索多 FPGA 架构上的大规模图处理。载于 2017 年 ACM/SIGDA 现场可编程门阵列国际研讨会论文集。217–226。[17] Jesse De Meulemeester、Antoon Purnal、Lennert Wouters、Arthur Beckers 和 Ingrid Verbauwhede。2023。{SpectrEM}：利用瞬态执行期间的电磁辐射。载于第 32 届 USENIX 安全研讨会（USENIX Security 23）。6293–6310。[18] John Demme、Robert Martin、Adam Waksman 和 Simha Sethumadhavan。2012。侧信道脆弱因子：衡量信息泄漏的指标。载于 2012 年第 39 届计算机体系结构年度国际研讨会（ISCA）。106–117。doi:10.1109/ISCA.2012.6237010[19] Eden Desmet、Suparna Kundu 和 Ingrid Verbauwhede。2026。乱序掩码：软件掩码密码学在乱序处理器上的侧信道泄漏。密码学电子印刷档案（2026）。[20] Colin Drewes、Tyler Sheaves、Olivia Weng、Keegan Ryan、Bill Hunter、Christopher McCarty、Ryan Kastner 和 Dustin Richmond。2024。开启、调谐并仔细聆听：最大限度地提升跨平台时间数字转换器中的侧信道恢复能力。ACM 可重构技术与系统汇刊 17，3（2024），30。[21] Eduardo Ferrufino、Luke Beckwith、Abubakr Abdulgadir 和 Jens-Peter Kaps。2023。[1]FOBOS 3：用于侧信道分析和基准测试的开源平台。载于《2023年硬件安全攻击与解决方案研讨会论文集》。5–14。[22] Hendra Guntur、Jun Ishii 和 Akashi Satoh。2014。侧信道攻击用户参考架构板卡 SAKURA-G。载于《2014年IEEE第3届全球消费电子会议（GCCE）》。IEEE，271–274。[23] Gregor Haas 和 Aydin Aysu。2022。Apple vs. EMA：针对苹果 CoreCrypto 的电磁侧信道攻击。载于《第59届ACM/IEEE设计自动化会议论文集》。247–252。[24] Dougall Johnson。 [n. d.]。Firestorm 概述。https://dougallj.github.io/applecpu/firestorm.html。苹果微架构研究；访问日期：2026年10月4日。[25] Nachiket Kapre 和 André DeHon。2009。使用 FPGA 加速 SPICE 模型评估。载于《2009年第17届IEEE现场可编程定制计算机研讨会》。IEEE，37–44。[26] Sagar Karandikar、Howard Mao、Donggyu Kim、David Biancolin、Alon Amid、Dayeol Lee、Nathan Pemberton、Emmanuel Amaro、Colin Schmidt、Aditya Chopra 等。2018。FireSim：公有云中的 FPGA 加速周期精确横向扩展系统仿真。载于《2018年ACM/IEEE第45届计算机体系结构年度国际研讨会（ISCA）》。IEEE，29–42。[27] George Karypis、Rajat Aggarwal、Vipin Kumar 和 Shashi Shekhar。1997。多级超图划分：在 VLSI 领域中的应用。载于《第34届年度设计自动化会议论文集》。526–529。[28] 是德科技。 [n. d.]。MSOX3104T 混合信号示波器：1 GHz、4个模拟通道加16个数字通道。https://www.keysight.com/s g/en/product/MSOX3104T/mixed-signal-oscilloscope-1-ghz-4-analog-16-digital-channels.html。访问日期：2026-10-02。[29] Pantea Kiaei、Zhenyuan Liu、Ramazan Kaan Eren、Yuan Yao 和 Patrick Schaumont。2021。Saidoyoki：评估流片前和流片后的侧信道泄漏。IACR密码学电子印刷档案。2021（2021），1235。[30] Donggyu Kim、Adam Izraelevitz、Christopher Celio、Hokeun Kim、Brian Zimmer、Yunsup Lee、Jonathan Bachrach 和 Krste Asanović。2016。Strober：适用于任意 RTL 的快速、精确的基于采样的能量仿真。ACM SIGARCH《计算机体系结构新闻》44，3（2016），128–139。[31] Andreas Kogler、Jonas Juffinger、Lukas Giner、Lukas Gerlach、Martin Schwarzl、Michael Schwarz、Daniel Gruss 和 Stefan Mangard。2023。{Collide+ Power}：利用基于软件的功耗侧信道泄漏不可访问数据。载于《第32届USENIX安全研讨会（USENIX Security 23）》。7285–7302。[32] Alexander Kropotov、Miquel Moreto 和 Behzad Salami。2026。EMiX：突破单 FPGA 的限制。arXiv:2604.27012 [cs.AR] https://arxiv.org/abs/2604.27012[33] Chester Lam。2024。Lion Cove：英特尔性能核的咆哮。https://chipsandcheese.com/p/lion-cove-intels-p-core-roars。访问日期：2026年10月4日。[34] Chester Lam。2025。深入了解 SiFive 的 P550 微架构。https://chipsandcheese.com/p/inside-sifives-p550-microarchitecture。访问日期：2026-09-30。[35] Chester Lam。2025。RISC-V 进展检查：P550 与 C910 基准测试。https://chipsandcheese.com/p/a-risc-v-progress-check-benchmarking。访问日期：2026-09-30。[36] Moritz Lipp、Andreas Kogler、David Oswald、Michael Schwarz、Catherine Easdon、Claudio Canella 和 Daniel Gruss。2021。PLATYPUS：针对 x86 的基于软件的功耗侧信道攻击。载于《2021年IEEE安全与隐私研讨会（SP）》。IEEE，355–371。[37] Jake Longo、Elke De Mulder、Dan Page 和 Michael Tunstall。2015。SoC it to EM：针对复杂片上系统的电磁侧信道攻击。载于《密码硬件与嵌入式系统国际研讨会》。Springer，620–640。[38] Edgar Mateos 和 Catherine H Gebotys。2010。侧信道的新型相关频率分析。载于《第5届嵌入式系统安全研讨会论文集》。1–8。[39] Mohammadmahdi Mazraeli、Yu Gao 和 Paul Chow。2023。面向数据中心的大规模多 FPGA 应用分区。载于《2023年第33届现场可编程逻辑与应用国际会议（FPL）》。IEEE，253–258。[40] Olivier Meynard、Denis Réal、Sylvain Guilley、Florent Flament、Jean-Luc Danger 和 Frédéric Valette。2010。频域中的电磁侧信道特征分析。载于《信息安全与密码学国际会议》。Springer，471–486。[41] Colin O'flynn 和 Zhizhang Chen。2014。ChipWhisperer：用于硬件嵌入式安全研究的开源平台。载于《构造性侧信道分析与安全设计国际研讨会》。Springer，243–260。[42] Kostas Papagiannopoulos、Ognjen Glamočanin、Melissa Azouaoui、Dorian Ros、Francesco Regazzoni 和 Mirjana Stojilović。2023。侧信道度量速查表。《ACM计算调查》55，10（2023），1–38。[43] Neha Prakriya、Yuze Chi、Suhail Basalama、Linghao Song 和 Jason Cong。2024。TAPA-CS：在分布式 HBM-FPGA 上实现可扩展的加速器设计。载于《第29届ACM计算机体系结构支持编程语言与操作系统国际会议论文集》。

编程语言与操作系统，第*3*卷。966–980。[44] Eliott Quéré、Maria Méndez Real、Alessandro Palumbo、Thomas Rokicki、Lilian Bossuet 和 Rubén Salvador。2026。SnoopyPower！针对缓存和一致性路径的远程功耗攻击。载于《*2026*年*IEEE*硬件导向安全与信任国际研讨会（*HOST*）》。IEEE，1–13。[45] Graham Schelle、Jamison Collins、Ethan Schuchman、Perrry Wang、Xiang Zou、Gautham Chinya、Ralf Plate、Thorsten Mattner、Franz Olbrich、Per Hammarlund 等。2010。使 Intel Nehalem 处理器核心可由 FPGA 综合实现。载于《第*18*届*ACM/SIGDA*现场可编程门阵列国际研讨会论文集》。3–12。[46] Falk Schellenberg、Dennis RE Gnad、Amir Moradi 和 Mehdi B Tahoori。2021。内部作案：针对 FPGA 的远程功耗分析攻击。*IEEE Design & Test* 38，3（2021），58–66。[47] Sebastian Schlag、Tobias Heuer、Lars Gottesbüren、Yaroslav Akhremtsev、Christian Schulz 和 Peter Sanders。2023。高质量超图划分。*ACM*实验算法学期刊27（2023），1–39。[48] David Spielmann、Ognjen Glamočanin 和 Mirjana Stojilović。2023。RDS：用于有效远程功耗分析攻击的 FPGA 布线延迟传感器。*IACR*密码硬件与嵌入式系统汇刊 2023，2（2023年3月），–。doi：。/tches.v。i。- Amisha Srivastava、Muskan Porwal 和 Kanad Basu。2026。CryptRISC：一种具有功耗侧信道防护的高性能密码学安全 RISC-V 处理器。*arXiv*预印本 *arXiv:2602.20285*（2026）。[50] Yaswanth Tavva、Rohan Juneja、Trevor E Carlson 和 Li-Shiuan Peh。2025。Ctscan：用于在边缘 CPU 上仿真功耗侧信道攻击的基于 CGRA 的平台。*ACM* 可重构技术与系统汇刊18，2（2025），1–36。[51] NewAE Technology。2025。CW310 Bergen板卡（Kintex FPGA目标平台）。访问日期：2025-02-09。https://rtfm.newae.com/Targets/CW310%20Bergen%20Board/ [52] NewAE Technology。2025。CW340 Luna板卡（OpenTitan基板）。访问日期：2025-02-09。https://rtfm.newae.com/Targets/CW340%20Luna%20Board/ [53] Russell Tessier、Kenneth Pocek 和 Andre DeHon。2015。可重构计算架构。*Proc. IEEE* 103，3（2015），332–354。[54] Shanquan Tian、Shayan Moini、Adam Wolnikowski、Daniel Holcomb、Russell Tessier 和 Jakub Szefer。2021。多租户 FPGA 中通用张量加速器上的远程功耗攻击。载于《*2021*年*IEEE*第*29*届现场可编程定制计算机年度国际研讨会（*FCCM*）》。IEEE，242–246。[55] Shengbo Tong、Haoyuan Li、Jiahao Xu、Chunyan Pei、Wenjian Yu、Shengjun Liu 和 Jian Shen。2024。Easypart：一种用于基于 FPGA 仿真的高效、全面的超图划分器。载于《第*43*届*IEEE/ACM*计算机辅助设计国际会议论文集》。1–9。[56] Nikita Veshchikov 和 Sylvain Guilley。2017。模拟器在侧信道分析中的应用。载于《*2017*年*IEEE*欧洲安全与隐私研讨会专题讨论会（*EuroS&PW*）》。IEEE，104–112。[57] Debao Wang、Yiwen Gao、Yongbin Zhou 和 Xian Huang。2024。重新审视针对复杂现代片上系统的现实电磁侧信道攻击。密码学电子印刷档案，论文2024/1322。https://eprint.iacr.org/2024/1322[58] Yingchen Wang、Riccardo Paccagnella、Elizabeth Tang He、Hovav Shacham、Christopher W Fletcher 和 David Kohlbrenner。2022。Hertzbleed：将针对 x86 的功耗{侧信道} 攻击转化为远程定时攻击。载于《第*31*届*USENIX*安全研讨会（*USENIX Security 22*）》。679–697。[59] Joonho Whangbo、Edwin Lim、Chengyi Lux Zhang、Kevin Anderson、Abraham Gonzalez、Raghav Gupta、Nivedha Krishnakumar、Sagar Karandikar、Borivoje Nikolić、Yakun Sophia Shao 等。2024。FireAxe：大规模 RTL 设计的分区式 FPGA 加速仿真。载于《*2024*年*ACM/IEEE*第*51*届计算机体系结构年度国际研讨会（*ISCA*）》。IEEE，501–515。[60] Yue Zha 和 Jing Li。2020。云中的 FPGA 虚拟化。载于《第二十五届计算机体系结构支持编程语言与操作系统国际会议论文集》。845–858。[61] Congwu Zhang、Panyu Wang、Yazhou Wang、Mingyu Chen、Yungang Bao、Yisong Chang 和 Ke Zhang。2025。HeteroProto：面向异构多 FPGA 片上系统原型验证的自动化 RTL 到比特流框架。载于《*2025*年现场可编程技术国际会议（*ICFPT*）》。IEEE，101–110。[62] Mark Zhao 和 G Edward Suh。2018。基于 FPGA 的远程功耗侧信道攻击。载于《*2018*年*IEEE*安全与隐私研讨会（*SP*）》。IEEE，229–244。[63] Dan Zheng、Xinshi Zang 和 Martin DF Wong。2021。Topopart：面向多 FPGA 系统的多级拓扑驱动分区框架。载于《*2021*年*IEEE/ACM*计算机辅助设计国际会议（*ICCAD*）》。IEEE，1–8。