Systematic Comparison and Improvement of Pre-silicon Leakage Analysis Tools

Abolfazl Sajadi
LIACS, Leiden University
Leiden, Netherlands
a.sajadi@liacs.leidenuniv.nl

Nusa Zidaric
LIACS, Leiden University
Leiden, Netherlands
n.zidaric@liacs.leidenuniv.nl

Todor Stefanov
LIACS, Leiden University
Leiden, Netherlands
t.p.stefanov@liacs.leidenuniv.nl

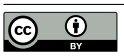
Nele Mentens
LIACS, Leiden University
Leiden, Netherlands
COSIC, ES&S, KU Leuven
Leuven, Belgium
n.mentens@liacs.leidenuniv.nl

Abstract

Side-channel attacks exploit physical information leakage, such as power consumption, to recover secret data, posing a serious threat to hardware security. To address these concerns early in the design cycle, pre-silicon leakage analysis is becoming a fundamental and mandatory step when building secure hardware.

The first part of this paper focuses on the systematic comparison of three notable pre-silicon side-channel leakage evaluation methodologies: RTL-PAT, PATCH, and ACA. The comparison is two-fold: (1) A qualitative comparison, based on a systemization of knowledge that decomposes the existing methodologies into three phases; (2) A quantitative comparison, focusing on the following metrics: runtime, and leakage localization granularity. For fairness, we run all experiments under identical conditions on a common benchmark: a 32-bit RISC-V core running a software implementation of Tiny-AES, and a hardware AES coprocessor, synthesized using a 22nm standard-cell library. The systematic comparison allows us to highlight the trade-offs between module-level vs. gate-level detection, i.e., coarse-grained vs. fine-grained leakage localization, and to identify tool-induced runtime overheads.

Based on these findings, the second part of this paper proposes an improved ACA-inspired methodology: ASSESS (Analysis of Side-channel Security through Enhanced pre-Silicon Simulations). It leverages a cycle-accurate toggle activity correlation, while bypassing the expensive time-based power extraction. The experimental results demonstrate that our approach preserves the fine-grained leakage localization of ACA while significantly reducing the analysis time. Our open-source methodology enables frequent leakage evaluations during the iterative ASIC design process. It empowers designers to pinpoint the leakage and apply targeted countermeasures before tape-out, reducing fabrication costs and strengthening hardware security.



CF Companion '26, Catania, Italy

ACM ISBN 979-8-4007-2569-2/26/05
https://doi.org/10.1145/3801488.3807896

Keywords

Side-Channel Leakage, Netlist Analysis, Power SCA, RISC-V, AES

ACM Reference Format:
Abolfazl Sajadi, Nusa Zidaric, Todor Stefanov, and Nele Mentens. 2026. Systematic Comparison and Improvement of Pre-silicon Leakage Analysis Tools. In *Proceedings of the 23rd ACM International Conference on Computing Frontiers (CF Companion '26), May 19–21, 2026, Catania, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3801488.3807896

1 Introduction

Digital hardware is the core of all modern technologies, ranging from cloud data centers to medical devices. Exploitable information leakage through side-channels has devastating consequences, such as privacy loss or even critical infrastructure failure. *Side-Channel Analysis* (SCA) can recover secrets, e.g., encryption keys, by observing physical artifacts such as power consumption, electromagnetic radiation, or execution time, rather than by breaking the underlying mathematics. Well-known power-analysis attacks on smart-card implementations of AES [5, 8, 19] and on standardized Trusted Platform Module (TPM) 2.0 security chips [13] extracted the keys in minutes, even when the cryptographic algorithms themselves were secure. Such incidents underscore the need to identify and mitigate side-channel vulnerabilities before chips are deployed. Nevertheless, the iterative process of analyzing chips after fabrication, and re-designing and re-fabricating them when side-channel weaknesses are detected, adds months and significant cost to the schedule, and fabricating an ASIC test chip solely for power analysis is rarely feasible. Therefore, there is a need to discover and neutralize leakage during the hardware design cycle (pre-silicon), not after fabrication (post-silicon). While post-silicon evaluation remains a final safeguard, it is reactive and expensive. Pre-silicon leakage detection enables engineers to insert targeted countermeasures, iterate quickly, and meet their goals while avoiding the need for broad, high-overhead countermeasures. However, pre-silicon leakage detection analysis has to overcome several challenges: (i) Adequate leakage localization granularity is needed to pinpoint which nets or gates leak, such that countermeasures remain local and low-overhead; (ii) The analysis must correlate closely with real silicon behavior, otherwise some leakage sources may not be discovered; (iii) Given the iterative nature of ASIC design, each

leakage detection analysis must be computationally efficient to support frequent design re-evaluation after design refinements and back-end adjustments.

Several existing ASIC-aware pre-silicon evaluation methodologies have already tried to address these challenges. Notable examples are RTL-PAT [16], which focuses on module-level leakage evaluation, and PATCH [20] and ACA [23], which focus on gate-level leakage evaluation. However, to the best of our knowledge, no prior work has performed a unified quantitative comparison of these methodologies using the same technology node and benchmarks. Consequently, the community lacks a clear understanding of the precise trade-offs not just qualitatively, but quantitatively between module-level and gate-level approaches. In addition, the existing methodologies have important limitations, i.e., RTL-PAT lacks fine-grained leakage localization, while PATCH and ACA incur prohibitive computational costs due to the reliance on expensive time-series power extraction.

Therefore, in this paper, we provide a unified quantitative comparison of these methodologies and propose an improved methodology for pre-silicon *Power Side-Channel* (PSC) leakage evaluation that addresses the aforementioned limitation by combining gate-level precision with the speed required to enable efficient iterative ASIC design. More specifically, the novel contributions of this paper can be summarized as follows:

1. We provide a **systemization of knowledge** by decomposing the RTL-PAT, PATCH, and ACA methodologies into distinct phases, namely Simulation, Power Estimation, and Leakage Assessment. This allows us to present a fair quantitative comparison under identical conditions, i.e., the same experimental setup, ASIC technology node (GF® 22FDX), and benchmark design (RISC-V/Ibex).
2. We develop a **common evaluation framework** by implementing and documenting practical refinements that improve the accuracy and/or runtime of each methodology. By making our framework and benchmarks available as open-source, we enable the research community to reuse this framework for future optimizations and evaluations.
3. Based on critical bottlenecks, identified during the aforementioned quantitative comparison, we design and implement a new open-source methodology called **ASSESS** (Analysis of Side-channel Security through Enhanced pre-Silicon Simulations). ASSESS is a cycle-accurate methodology that supports gate-level leakage localization, hence addressing challenge (i). To address challenge (ii), our methodology requires one run with a commercial power estimation tool using a targeted ASIC standard-cell library. addressing challenge (iii), ASSESS eliminates the expensive time-series power extraction by cycle-accurate analysis of the nets' switching activity within the design.

This paper is organized as follows. Section 2 briefly reviews related work. Section 3 presents an overview of the three previously proposed methodologies selected for comparison. Section 4 compares these methodologies and experimental results, and offers improvement suggestions. Section 5 details our proposed methodology and examines its outcomes. Finally, Section 6 concludes the paper.

2 Related Work

Side-channel vulnerabilities can be detected at any design level of abstraction, from detailed transistor-level simulations to higher-level gate and RTL analyses where the choice of abstraction level strongly affects both the detection accuracy and runtime. At the transistor level, some authors rely on circuit simulation with SPICE to estimate the power consumption and identify potential sources of leakage [17]. Although this approach offers high accuracy, it also requires considerable computational resources and scales poorly for larger designs.

Saidoyoki *et al.* [6] introduces a dual-setting evaluation platform that compares pre-silicon leakage estimations with post-silicon measurements, demonstrating that early-stage simulations can achieve fine-grained localization resolution and robust root-cause analysis. Their work highlights the necessity of leakage analysis at multiple levels of abstraction (from RTL to layout).

Still at the RTL level, PSC-TG [24] uses test-pattern generation and information flow tracking to detect worst-case power side-channel leakage before fabrication. By identifying sensitive variables and generating inputs that maximize leakage, PSC-TG allows early, cost-effective countermeasures without requiring an actual silicon prototype. The successor of PSC-TG, RTL-PAT (formerly RTL-PSC) [16], also performs a side-channel leakage assessment at the RTL stage, offering a comparatively fast and technology-agnostic solution. Its emphasis on module-level analysis nevertheless overlooks finer-grained factors such as glitches, which are known sources of side-channel leakage [12], and disregards placement and routing details. Designers therefore cannot always determine which specific nets or signals dominate the leakage, limiting the accuracy of any subsequent countermeasures. SCAR [22] applies a graph neural network to RTL control–data flow graphs to flag "leaky" code and then uses large language models (LLM) to auto-patch the code. Their approach depends on manually labeled "leaky" modules and does not capture post-routing effects (e.g., glitches and coupling). The scalability of LLM-generated mitigations remains an open challenge.

Moving further down in the design flow abstraction levels, several methodologies operate directly on gate-level netlists. PATCH [20] ranks nets by correlating each net's toggling activity with the instantaneous total power trace. Since the analysis is restricted to pre-selected *sensitive windows*, glitch-induced leakage outside those windows can remain undetected. Architecture Correlation Analysis (ACA) [23] achieves even finer localization granularity by correlating the simulated power of every gate with a hypothesized leakage model. Yao *et al.* [7] applied ACA to both a stand-alone AES coprocessor and a RISC-V SoC, showing that only a few percent of gates account for the bulk of the observable leakage. The gate-level visibility comes at a cost: extracting and analyzing thousands of time-series power traces required by ACA dominates the simulation runtime on SoC-scale designs. In general, both PATCH and ACA depend on accurate time-series of power data from commercial EDA tools, which become increasingly expensive (in time and effort) to generate when the design size and process complexity increase. PathFinder [10] builds on ACA by using ACA-based localization to select high-leakage logic cones (i.e., gate-level region spanning from primary inputs to primary outputs) before inserting

logic-level obfuscations, so any speed or accuracy improvements to ACA directly benefit PathFinder. Since our flow also builds on ACA, our results will also improve PathFinder. Bhasin et al. [2] generate design-time SCA traces via SPICE/FAST-SPICE or timing-annotated VCD (Value Change Dump) with equal-weight transitions. However, they do not provide per-gate Leakage Impact Factors (LIFs), so the approach lacks gate-level leakage localization. Unlike Count Your Toggles [18], which focuses on leakage estimation using aggregate switching activity, our methodology performs cycle-accurate, gate-level leakage localization with per-gate LIF ranking while remaining technology-aware.

Another line of work operates at the post-placement stage. Karna [21] adjusts the power profile by swapping gates. This can reduce leakage when sufficient positive timing slack is available, but that requirement restricts design flexibility and does not directly reveal which nets are the most leakage-prone.

Our goal differs: we target gate-level, technology-aware, label-free leakage localization with per-gate LIFs and reproducible timing. Hence, we do not adopt SCAR's ML/auto-patch flow. Instead, our method quantifies leakage on the ASIC fabric that will actually be fabricated rather than proposing edits at RTL. The remainder of this paper focuses on the reconstruction, comparison and improvement of three notable pre-silicon PSC leakage assessment methodologies, that operate on different levels of abstraction, from RTL down to placed gate-level netlists, namely RTL-PAT, PATCH, and ACA.

3 RTL-PAT, PATCH and ACA in Three Phases

In this section, we examine the three main phases common to all three methodologies, namely simulation, power estimation, and leakage assessment, shown at the top part of Fig. 1, 2 and 3. This decomposition was already used by the authors of the RTL-PAT methodology [16], and we apply it to all methodologies examined in this paper. We represent the three existing methodologies graphically for easier comprehension of subtle differences. The comparison in this section focuses on the utilized leakage models and the leakage localization granularity the methodologies can achieve. Whenever a methodology requires time-series power traces, we assume that these traces are obtained by using a *commercial power tool (CPT)* e.g., Synopsys PrimeTime PX or Cadence® Joules.

3.1 RTL-PAT: PSC Leakage Assessment at RTL

RTL-PAT [16] is a pre-silicon SCA methodology that operates directly at the RTL design. It statistically analyzes differences in module-level switching activities, enabling designers to efficiently identify leakage-prone modules in the design. Fig. 1 outlines the main phases of RTL-PAT.

Simulation. A common SCA practice is to select two keys with the maximum Hamming distance (HD) (typically 0x00...00 and 0xFF...FF) to maximize the statistical separation between the resulting power distributions. This choice is made under the assumption that the plaintext is XORed with the key, and two keys with a larger Hamming distance are likely to produce more distinct switching activities. RTL simulations are executed separately for each key using the same set of N random plaintexts, and the signal transitions are recorded into VCD files that capture the switching activities.

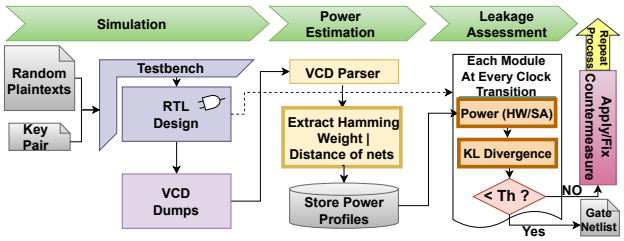


Figure 1: The RTL-PAT methodology, based on Fig. 1 in [16].

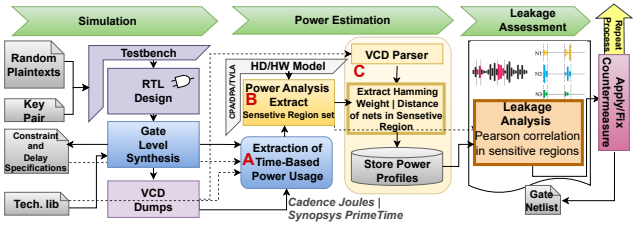


Figure 2: The PATCH methodology, based on the description in [20].

Power Estimation. The generated VCD files in the first step are parsed to estimate dynamic power consumption using two common models. The first model, Hamming Weight (HW), estimates the power consumption as proportional to the number of nets at logic '1', based on a simplified CMOS characteristic where storing or driving a logic '1' is assumed to incur higher power than a logic '0'. The second model, HD, estimates the power consumption as proportional to the number of net transitions, either from '0' → '1' or '1' → '0', based on the assumption that the transitions cause dynamic power consumption.

Leakage Assessment. To assess how easily an attacker could distinguish two secret keys, RTL-PAT compares the simulated power-consumption distributions produced by each key. The chosen statistical estimator for this comparison is the *Kullback–Leibler* divergence (D_{KL}). The larger D_{KL} is, the more distinguishable the two keys are and the more vulnerable the design becomes. Assuming the two distributions X and Y are Gaussian with means μ_X, μ_Y and variances σ_X^2, σ_Y^2, the closed-form D_{KL} expression is given as:

$$D_{\mathrm{KL}}(X \parallel Y) = \frac{(\mu_X - \mu_Y)^2 + \sigma_X^2 - \sigma_Y^2}{2\sigma_Y^2} + \ln\left(\frac{\sigma_Y}{\sigma_X}\right) \tag{1}$$

RTL-PAT flags every module whose D_{KL} score exceeds a user selected threshold. The original work tabulates thresholds in terms of an *adversary failure probability* P_r (Table 1). For example, keeping $D_{\mathrm{KL}} < 0.03$ guarantees $P_r \geq 0.9$, i.e., the attacker has at least a 90% chance of guessing the wrong key [16]. Modules that violate the chosen bound are marked *leaky*. Designers can then apply countermeasures to the leaky modules only, thereby avoiding unnecessary area or timing overhead caused by the countermeasures.

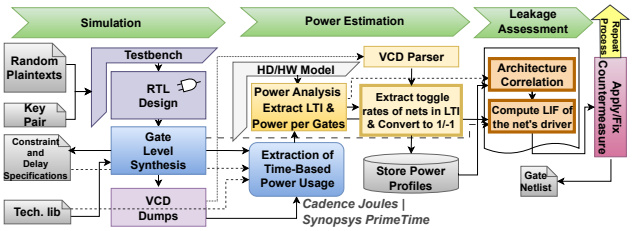


Figure 3: ACA methodology, based on the description in [23].

3.2 PATCH: Gate-Level PSC Leakage Assessment & Localization

PATCH is a gate-level methodology for pinpointing PSC vulnerabilities, using the netlist of a design, by correlating each net's toggling activity with a HD/HW power model. Operating at the gate-level abstraction, PATCH relies on toggling statistics that inherently incorporate real gate delays and glitches. Fig. 2 illustrates PATCH w.r.t. the three aforementioned phases.

Simulation. A gate-level netlist is obtained from the synthesized design, together with technology-specific constraints (SDC/SDF files). The design is then simulated with multiple plaintext inputs to produce VCD files, accurately accounting for timing delays.

Power Estimation. First, a CPT leverages the netlist, timing data, and technology libraries to generate time-series power traces for every VCD file (step *A* in Fig. 2). Next, leakage analysis techniques such as Differential Power Analysis (DPA), Correlation Power Analysis (CPA), or T-tests [11] detect time windows, called *sensitive regions*, during which the design is most vulnerable (step *B*). Finally, a dedicated VCD parser tracks toggles on every net in the sensitive regions, estimates net-level dynamic power consumption using a HD/HW model, and stores the power profiles (step *C*).

Leakage Assessment. Within the sensitive regions, the net-level power traces are individually correlated against the total design power consumption to identify nets that most significantly contribute to leakage. A high correlation implies that modifying these nets' switching behavior or applying localized countermeasures may substantially reduce side-channel attack vulnerability. Designers can then selectively harden the nets with high correlation to meet their acceptable leakage level.

3.3 ACA: Architecture Correlation Analysis

ACA is a design-time methodology intended to identify which gates in a gate-level netlist contribute the most to side-channel leakage. Three inputs are required: (1) a gate-level netlist, (2) a known leakage model $L(V)$ where the leakage model L is a function computed over a secret intermediate variable V, and (3) time-series power traces obtained from stimuli that exercise the secret variable V. Once these are available, ACA proceeds with the three main phases shown in Fig. 3.

Simulation. Similar to PATCH, a gate-level simulation is performed on the netlist (along with SDF/SDC files). Various plaintexts or input stimuli are applied, and each net's toggling activity is recorded in VCD files.

Power Estimation. After generating power traces using a CPT, ACA determines a *Leakage Time Interval (LTI)* by correlating the total power consumption $P(t)$ with the leakage model $L(V)$ as:

$$\rho_{L(V),t} = \frac{\mathrm{cov}\big(L(V), P(t)\big)}{\sigma_{L(V)}\,\sigma_P}, \tag{2}$$

where cov is the covariance operation, and $\sigma_{L(V)}$ and σ_P are the standard deviations of $L(V)$ and P, respectively. Any time window in which $\rho_{L(V),t}$ exceeds the Pearson correlation threshold (as defined under the "Pearson correlation Threshold" section of Table 1, depending on the chosen confidence level and sample size) is treated as containing meaningful leakage and constitutes the LTI. While determining the LTI, the CPT also provides gate-level average power data for subsequent analysis.

Leakage Assessment. Within each LTI, ACA computes an *architecture correlation factor* C_i for every net i. Let K_i be the *toggle trace* of net i, with +1 if the net toggles and −1 otherwise, and the toggle trace of the leakage model is H. The correlation factor is computed as the dot product (Eq. 3) – see Table 4 for a simple example using stimuli S_0–S_3. Subsequently, the *LIF* F_i is defined by scaling C_i with the gate's average power consumption in the LTI (Eq. 4).

$$C_i = K_i \cdot H \tag{3}$$

$$F_i = C_i \frac{P_i}{P_T} \tag{4}$$

Here, P_i is the average power consumption of the gate driving net i within the LTI, and P_T is the design's total power over that interval. Larger F_i flags gates whose toggling aligns with the model and that draw significant power. ACA ranks gates by F_i to prioritize and apply countermeasures.

4 Common Evaluation Framework and Comparison.

To compare the three pre-silicon methodologies, introduced in Section 3, under identical conditions, we built a moderately complex benchmark around the open-source Ibex 32-bit RISC-V core [4]. Ibex is written in SystemVerilog and supports the RV32I/E, M, C and B extensions. A software implementation of AES-128 (*Tiny-AES* [9]) executes on the core and generates the switching activity within the core hardware structure that is analyzed by each methodology. The Ibex design was synthesized using the GF® 22FDX technology (Genus 21.14) and contains ≈8K gates which are driving nets, with 10 MHz clock frequency. In the following, we compare how long each methodology takes to analyze the Ibex design, i.e., its computational runtime overhead.

We implemented the three methodologies in Python[1] and their analysis phases were executed on the same workstation (Intel Core i7, 8 cores, 32 GB RAM) to ensure fair comparison of the methodologies' runtime and memory requirements. Since the power trace generation is both multi threaded and memory intensive, it was carried out on a separate server equipped with an AMD Ryzen 9 3900X (12 cores, 128 GB RAM) and a large local storage. The power traces were sampled every 25 ns, i.e., four samples per 100 ns clock period, matching the timing resolution required by all methodologies. Power estimates were obtained with CPT (Cadence® Joules) using the GF 22FDX standard-cell library, while side-channel analyses were carried out with the ChipWhisperer Analyzer [14]. Using

[1]The source code and scripts available at: https://github.com/abolfazlsajadi/ASSESS

Table 1: Thresholds for statistical estimators used by RTL-PAT [16] and ACA [23].

KL Divergence Threshold				Pearson Correlation Threshold			
P_r	KL	P_r	KL	Confidence Interval	n=600	n=1000	n=2000
> 0.96	< 0.01	> 0.61	< 0.50	99%	±0.105	±0.081	±0.058
> 0.90	< 0.03	> 0.45	< 1.12	95%	±0.080	±0.062	±0.044
> 0.80	< 0.12	> 0.32	< 2.00	90%	±0.067	±0.052	±0.037

these CPT generated traces, a correlation power analysis (CPA) under the Hamming-Weight leakage model revealed all 16 Tiny-AES key bytes with only 45 traces [11, 14].

4.1 The RTL-PAT Methodology on Ibex

The RTL-PAT methodology, described in Section 3.1, requires two cryptographic keys to compare their switching activity distributions. A naive all-zero vs. all-one pair, as suggested in the original paper [16], cannot be used on Ibex, since the multiply/divide unit detects a division by zero in the first cycle and terminates execution. To preserve a high Hamming distance and constant-time execution, we instead use the two fixed key patterns `00 FF 00 FF. . .` and `FF 00 FF 00. . .` . For each key, we record 500 VCD files (by QuestaSim). Each trace covers 10K clock cycles sampled twice per cycle, yielding 20K time samples per trace for the leakage evaluation.

Fig. 4 depicts the 37 Ibex modules, grouped into plots by their D_{KL} divergence, and reports their cycle-by-cycle leakage. Because RTL-PAT analyzes only entire modules, it cannot pinpoint which nets inside a module are responsible for the leakage, and a large D_{KL} value does not automatically imply an exploitable vulnerability. Table 2 shows a breakdown of the RTL-PAT runtime. The overwhelming bottleneck is the VCD parsing. However, parallelizing the gathering transitions stage (VCD parsing) reduces the end-to-end run time from almost six hours to about two hours. The D_{KL} divergence computation itself requires only a few seconds. Although Ibex is smaller than a full multi-core SoC, it is a realistic synthesized RISC-V core that serves as a common benchmark for all three methodologies and for our RTL-PAT optimization. However, because RTL-PAT detects leakage only at the coarse granularity of modules, we next turn to gate-level techniques that can localize vulnerable logic more precisely.

4.2 The PATCH Methodology on Ibex

Unlike RTL-based approaches, PATCH leverages gate-level information. Fig. 5 shows the CPA results, with the highest correlation for each key byte highlighted. The x-axis denotes the sample index;

Table 2: RTL-PAT end-to-end runtime (500 × 2 traces).

Phases	Step	Per trace / once	Total
Sim.	Vmem gen.	141ms	2m 24s
	VCD dump	4.43s	1h 16m
Pwr. Est.	VCD parsing[1]	16.6s	4h 36m
	VCD parsing[2]	2.39s	41m 39s
Lkg.A.	D_{KL} divergence	3.13s	3.13s
end-to-end runtime[1]			5h 54m
end-to-end runtime[2]			2h 6s

1: single-thread 2: multi-thread (16)

traces are sampled every 25 ns (four samples per 10 MHz clock cycle). We first identify *sensitive regions*, i.e., indices with significant leakage, and expand each by ±20 neighbouring samples (≈±5 clock cycles) to form *sensitive regions*. Within these regions, we analyze the net switching activity and compute the Pearson correlation against the HW model.

Fig. 6 shows the distribution of absolute Pearson correlations between each net and the HW model for all 16 key bytes. The x-axis groups $|r|$ into four ranges, and the y-axis (log scale) reports how many nets fall into each range. Each bar corresponds to one key byte index. For all key bytes, the vast majority of nets have low correlation ($|r| < 0.5$), while only a small fraction reach the highest bin ($0.8 \leq |r| \leq 1.0$). Across the ~ 8k nets in Ibex, fewer than 10^3 nets exceed $|r| > 0.8$, so PATCH reduces the set of nets that require detailed inspection from the full design to this much smaller subset of highly correlated candidates.

Table 3 shows a breakdown of the PATCH runtime. The dominant cost is the power extraction with CPT (about 72 days), followed by gate-level simulation ≈ 1 *hour* and the original net-activity parsing ≈ 13 *hours*. According to the PATCH reference flow (Fig. 2), the net activity is usually extracted from the entire VCD file before isolating sensitive regions, which becomes memory-intensive for large designs and prevents multi-threading. To optimize this, we extract the net activity only within the sensitive regions, reducing the required memory by ≈ 97%. Without region reduction, processing the full design requires 78.7 s/trace. When restricted to the reduced

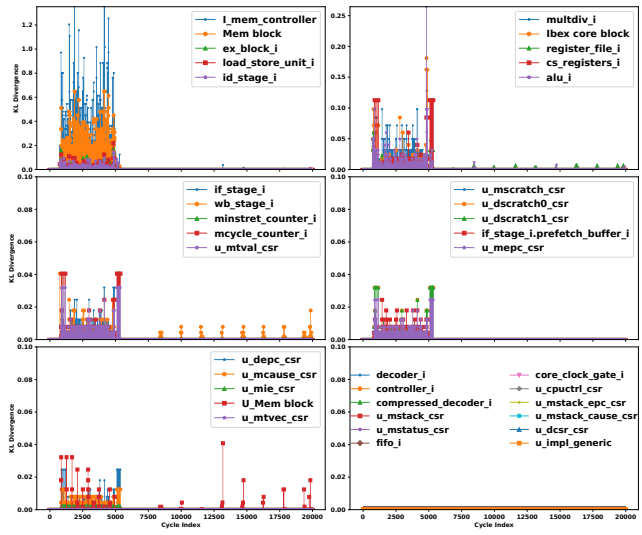


Figure 4: Results of the RTL-PAT method on Tiny-AES running on an Ibex core.

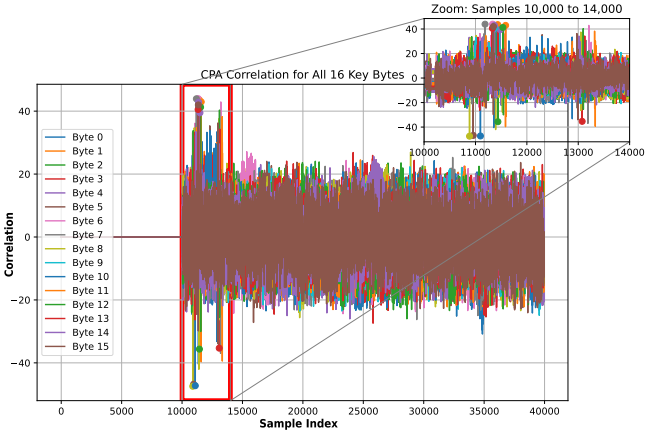


Figure 5: CPA per key byte (600 traces) with peaks marking sensitive moments.

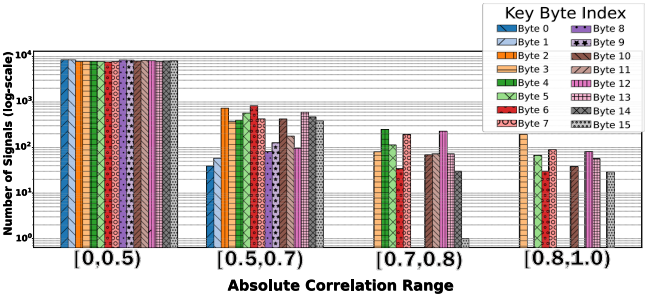


Figure 6: Distribution of signal correlations with the HW-based key bytes.

Table 3: PATCH end-to-end runtime (600 traces).

Phases	Step	Per trace	Total
Sim.	Vmem gen.	141ms	1m 24s
	Synthesis (once)	–	5m 53s
	VCD dump	6.32s	1h 03m
Pwr. Est.	Power extraction	2h 54m 26s	72d 15h 36m
	VCD parsing	78.7s	13h 7m
Lkg.A.	Pearson's ρ (16B)	23.4s	3h 53m 36s
end-to-end runtime			**73d 6h 22m**

sensitive regions, the runtime decreases to 16.4 s/trace using a single thread and further to 0.86 s/trace with 16 threads, resulting in an overall time saving of $\approx$ 13 *hours* hours across 600 traces.

The authors of PATCH evaluated a fully unrolled AES implementation (coprocessor) [15] but did not disclose the technology node. Our experiments with a 22nm library did not result in a similar timing as theirs, even though the design was comparable. The difference most likely stems from different technology libraries, delay models, sampling resolutions, or EDA tools (for CPT they used PrimeTime PX; we used Joules). In our own experiments, we quantified this effect by synthesizing the same fully unrolled AES coprocessor [15] using two standard-cell libraries that differ only in technology node, thereby illustrating the impact of the chosen/target technology on the runtime of this methodology.

- With a 0.18 μm library, synthesis finished in 120 s and CPT needed 1 min 04 s to generate a single power trace.

Table 4: Example of Architecture Correlation

	S0	S1	S2	S3	C_{ij}
Leakage Model (H_j)	+1	−1	−1	+1	
net0 (K_0)	+1	−1	−1	+1	4
net1 (K_1)	+1	+1	+1	+1	0
net2 (K_2)	−1	−1	+1	−1	−2

- Retargeting the identical RTL and tool settings to a 22nm library stretched synthesis time to 811 s, while power trace generation ballooned to 54 min 44 s per trace.

The order-of-magnitude slow-down can mainly be explained by the richer timing and power-characterization formats used at advanced nodes which drive both larger libraries and finer-grained delay/power interpolation within CPT. Consequently, it is important to report the technology node alongside the design complexity and sampling rate when comparing power analysis runtimes for hardware side-channel studies.

Because the leakage model compares the activity of individual nets against byte-level differences and does not account for the net's physical location or the identity of its driving gate, some false positives remain. That is, certain nets, such as primary inputs, may appear correlated with the leakage model despite having minimal impact on the overall power consumption. Nevertheless, this approach significantly narrows down the candidate set compared to blanket countermeasures. This motivates a finer-grained analysis in the form of a per-gate ranking method known as ACA, which we examine next.

4.3 The ACA Methodology on Ibex and AES coprocessor

In this section, we evaluate the performance of ACA not only on Ibex running Tiny-AES but also on a stand-alone hardware module, namely an AES coprocessor. This extra evaluation enables more thorough comparison of ACA with our ASSESS methodology presented in Section 5. The AES coprocessor implements AES-128 with on-line key expansion and completes one encryption in 11 clock cycles. The coprocessor design does not include dedicated side-channel countermeasures and is available as an RTL Verilog implementation [3].

Methodology and leakage model. Following the original ACA study [23] (summarized in Section 3.3), we adopt the HD leakage model, where each elements corresponds to the HD between two consecutive values of the 128-bit Tiny-AES state register. Fig. 7(1) shows a zoom-in on the first S-box execution and the corresponding Pearson correlation (Eq. (2)), computed from an initial set of 600 input patterns. The dominant peak clearly identifies the prospective LTI, with a correlation magnitude of $\rho_{L(V),t} \approx 0.14$, which exceeds the 99 % confidence threshold of 0.105 (red line) for 600 traces (Table 1). This confirms both the suitability of the HD model and the correctness of the chosen LTI. Fig. 7(2) shows the Pearson correlation for the 11-cycle AES coprocessor (including two additional cycles for data loading and result readout). With 1500 traces, we obtained $\rho_{L(V),t} \approx 0.13$, which is above the 99 % confidence threshold of 0.069 (red line). **Deriving the LIF distribution.** After recording the toggle trace for every net i within each LTI

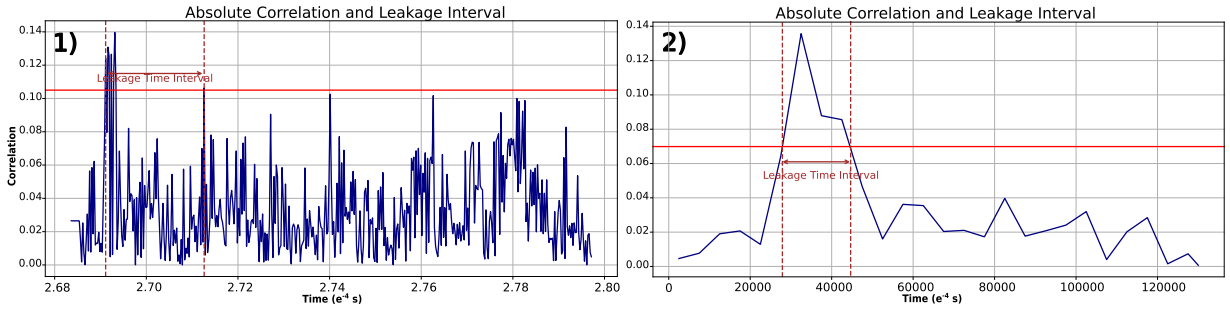


Figure 7: LTI obtained for first S-box Tiny-AES state bit 0 correlation 1) using 600 traces on Ibex (zoomed-in); 2) using 1500 traces on AES coprocessor.

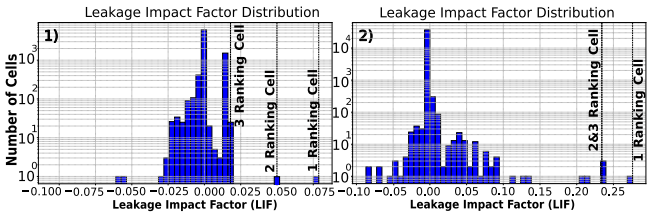


Figure 8: ACA LIF distribution for 1) 600 traces on Ibex; and 2) 1500 traces on AES coprocessor (log scale).

(Section 3.3), we compute the *architecture correlation* C_i with the dot product of Eq. (3); this measures the functional dependency between net i and the leakage model H. A large Pearson coefficient in Eq. (2) indicates that the *total* power follows the model, whereas a large C_i pinpoints the *individual* net that responsible for it. Following [23], we traverse the netlist to locate the gate that drives each net i. Each drivers average power P_i is obtained from the CPT and then normalized by the design-wide mean power P_T to obtain the *leakage-impact factor* F_i in Eq. (4). The gates with the highest F_i contribute the most to side-channel leakage, which enables ACA to rank their relative impact on the overall power-leakage footprint.

Fig. 8 illustrates the LIF distribution for all gates in both the Ibex design executing Tiny-AES (part 1) and the AES coprocessor (part 2), as obtained from the ACA output. In both cases, the distribution is strongly skewed: only a small fraction of gates exhibit large LIF values, meaning that a few gates dominate the side-channel leakage predicted by the chosen model. The results show that the top-ranked element in both cases is the AES state bit 0: a flip-flop in the Ibex register file and a flip-flop in the coprocessors internal state for AES block. In both designs, these registers toggle after every AES round, making them the most active and consequently the leakiest components. The next highest entries correspond to gates within the S-box logic directly driven by the state register outputs. Beyond these, the LIF values drop sharply, indicating that the remaining gates contribute only marginally to the overall side-channel leakage.

Runtime of the reference ACA flow. Table 5 reports the end-to-end runtime for the ACA methodology of [23]. The dominant bottleneck of ACA is the power extraction step performed with CPT, which generates full gate-level power traces. For the Ibex core, this step requires approximately 2.9 hours per trace, resulting in a total runtime of about 72 days for 600 traces. All other phases of the methodology (correlation, VCD parsing, and netlist analysis) together add only about three additional hours. In contrast, for the AES coprocessor, the total power extraction time is significantly shorter: roughly 18 seconds per trace (7.5 hours total for 1500 traces), owing to the lower number of clock cycles (two samples per clock).

Sampling-rate versus runtime. In our experiments, we used CPT (Cadence Joules) with a sampling rate of four samples per clock cycle (25 ns/sample on a 100 ns period) for the Ibex core and two samples per clock cycle (5 ns/sample on a 10 ns period) for the AES coprocessor, ensuring that glitches and intra-cycle delays are fully captured. Even when using a lower sampling rate, the power extraction step remains the dominant contributor to the total runtime. The timing results for Ibex are as follows:

- 2 samples/cycle: 1 h 14 min per trace ⟶ 31 days for 600 traces;
- 1 sample/cycle: 29 min per trace ⟶ 12 days for 600 traces;
- 1 sample/2 cycles: 16 min per trace ⟶ 7 days for 600 traces.

Thus, although accuracy can be traded for speed, power extraction remains the dominant bottleneck in ACA—particularly when frequent back-end revisions (e.g., new technology corners) necessitate a full re-run.

Comparison and key observations. The last three columns in Table 7 summarize the main features of the RTL-PAT, PATCH, and ACA methodologies. RTL-PAT operates at a higher level of abstraction (RTL), compared to PATCH and ACA. This leads to much shorter end-to-end runtime, e.g., ≈ 2 hours for the Ibex benchmark, and eliminates dependencies on technology libraries and CPTs. However, the shorter runtime comes at the cost of a coarse leakage localization granularity (i.e., leakage of modules) and a high probability of undiscovered leakage sources. In contrast, PATCH and ACA achieve medium to fine localization granularity (i.e., leakage of nets/gates) by taking into account nets/gates delays (i.e., delay awareness). ACA is the most effective methodology compared to RTL-PAT and PATCH because it pinpoints not only the nets causing leakage, but also identifies the driving gates and assigns LIF to each gate to quantify its individual contribution. This high effectiveness of ACA comes with a very high end-to-end runtime, e.g., more than 72 days for the Ibex benchmark. The main reason for this is the time-consuming power extraction step (Table 5), which is sensitive to the technology node, design complexity, and design latency in terms of clock cycles. Such a long step, in an iterative ASIC design flow, is not acceptable in cases where short time-to-market is required. This motivates us to develop the ASSESS methodology, presented in

Table 5: ACA runtime breakdown for the Ibex and AES coprocessor.

Phases	Step	Ibex (600 traces)		AES coprocessor (1500 traces)	
		Per-trace	Total	Per-trace	Total
Sim.	Vmem generation	141 ms	1 m 24 s	—	—
	Synthesis (once)	—	5 m 53 s	—	36 m 9 s
	VCD dump	6.32 s	1 h 3 m	3 s	1 h 15 m
Pwr. Est.	Power extraction	2 h 55 m	72 d 15 h	18 s	7 h 30 m
	Power analysis (HD)	—	13 s	—	10 s
	VCD parsing	2.5 s	25 m	3.47 it/s	7 m 12 s
	Per-gate power scan (Joules)	—	19 s	—	2 m 49 s
Lkg.A.	Net-list parsing & LIF	40 nets/s	3 m 28 s	12.5 nets/s	50 m 12 s
end-to-end runtime			**72 d 15 h 31 m**		**10 h 22 m**

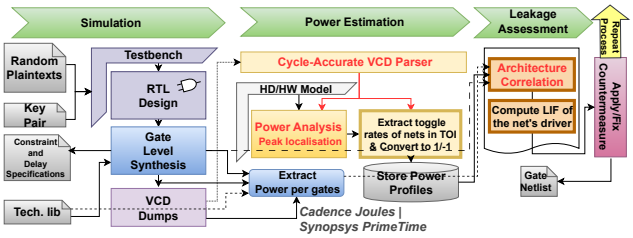


Figure 9: The ASSESS methodology (this work).

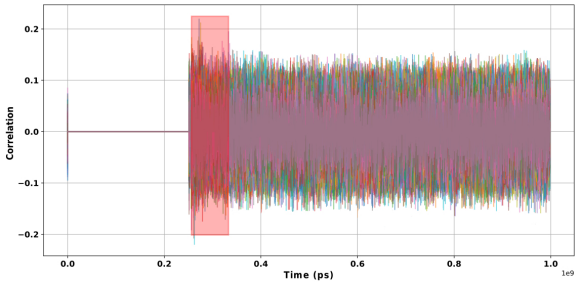


Figure 10: Correlation of total toggles with 128 HD models; TOI highlighted.

Section 5, which delivers the ACA's leakage localization granularity with the end-to-end runtime of RTL-PAT.

5 ASSESS: Proposed Methodology

Motivated by iterative ASIC design flow, we developed a leakage estimator that is both fast and gate-level leakage localization granularity. The ACA methodology namely burns *days* to obtain the time-series power traces (cf. Table 5); such long back-end runs are unacceptable during an iterative design flow.

Our fast PSC methodology **ASSESS** (Analysis of Side-channel Security through Enhanced pre-Silicon Simulations), shown in Fig. 9, retains the gate-level accuracy of ACA, yet finishes in <2 h on the same 600-trace data set for the *Ibex* case study (**a 960× speed-up**). The key idea is simple: *never simulate time-series power traces.* With runtime reduced from days to a coffee break, ASSESS provides the information needed to address vulnerabilities before the tape-out.

Method overview. Similar to the classic ACA methodology, ASSESS also follows three phases. We keep the *Simulation* phase (RTL to gate synthesis followed by a conventional VCD dump) unchanged, and the final *Leakage-assessment* phase only slightly modified. The main difference lies in the *Power-estimation* phase. We employ a two-pass procedure, at the heart of which lies a cycle-accurate VCD parser: the first pass is used for the peak localization, and the second pass for processing cycle-accurate VCD traces (highlighted red in Fig. 9).

Power estimation phase - Pass 1 → Peak localization. In this step, we aim to identify time windows during which the design exhibits stronger leakage. Isolating such windows significantly reduces the search space and conserves computational resources. Rather than relying on precise and computationally expensive bounds, such as those used in LTI-based approaches like the classic ACA flow, we look for time windows that exhibit high correlation with each model in a pre-defined set of possible leakage models. The results derived from this step do not directly determine the leakage model but help to limit the analysis to the most promising time regions.

To achieve this, we use the results from the Cycle-Accurate VCD Parser and calculate the total number of signal toggles across all nets at each clock edge. During peak localization we compute the Pearson correlation between the obtained toggle vector and each of the 128 HD models for Tiny-AES. The clock cycle(s) showing the highest correlation define our *Time Of Interest* (TOI), as illustrated in Fig. 10, where the TOI is highlighted in red. Processing the complete set of traces takes ≈ 33 minutes for Ibex core, This step is omitted for AES coprocessor due to its significantly smaller number of cycles.

Robustness to the threshold. A practical nuisance of the classic ACA flow is the manual choice of LTI: shift the Pearson threshold a little and the set of cycles classified as "leaking" changes, which may reorder the final LIF list. Our parser works *cycle-by-cycle*: every clock edge is evaluated individually, no binary decision "toggle inside/outside the LTI" is required, hence the ranking is insensitive to how wide a window the designer selects. The benefit of a compact window such as TOI is the runtime: a smaller slice of the data is processed, but correctness is not affected.

Power estimation phase - Pass 2 → Cycle-Accurate VCD Traces. Inside the TOI, we revisit the VCD and construct one binary toggle trace per net: $K_i(t) \in \{-1, +1\}$, with $K_i(t) = +1$ if net i toggles at cycle t, and -1 otherwise. For the Ibex core, which contains approximately 8.3k nets, the parser processes each nets in about

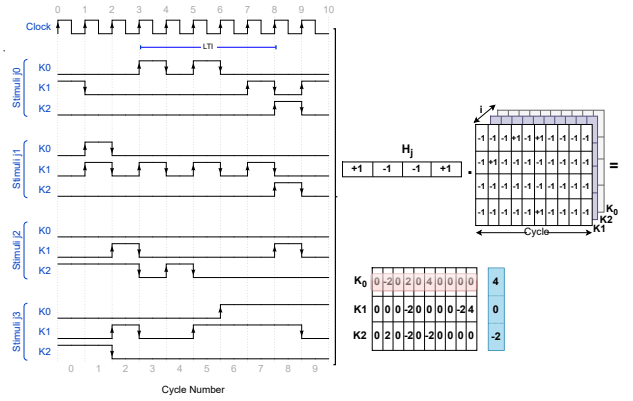


Figure 11: Cycle-accurate architecture correlation Eq. (5) inside the TOI. Net K_0 clearly realizes the leakage model.

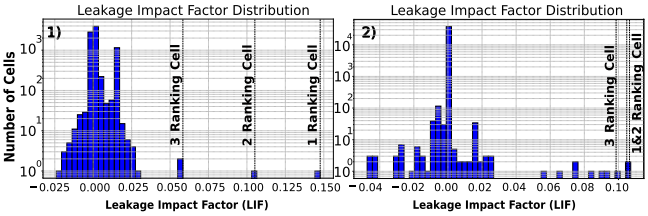


Figure 12: ASSESS LIF distribution for 1) 600 traces on Ibex; and 2) 1500 traces on AES coprocessor (log scale).

18 ms, resulting in a total runtime of around 2.5 minutes for the selected time window (Tab. 6). For the AES coprocessor, which includes roughly 38k nets, the total parsing time is around 2.5 minutes, mainly due to its much smaller number of analyzed cycles.

Leakage assessment phase - Architecture correlation. For every clock cycle t we form the dot product between the leakage-model toggle vector $\mathbf{H}$ and the toggle vector $K_i(t)$ of net i obtained from the cycle-accurate VCD parser, as $C_i(t) = K_i(t) \cdot \mathbf{H}$. A large $C_i(t)$ means net i follows the leakage model in cycle t and is thus a likely source of leakage. Given that $K_i \in \{-1, +1\}^{j \times \text{clk}}$ and $\mathbf{H}_j \in \mathbb{R}^{j \times 1}$, the results are $K_i \mathbf{H}_j \in \mathbb{R}^{1 \times \text{clk}}$ where j is the number of stimuli, clk the number of clock edges, and i the total number of nets. We show this calculation in Eq. (5).

$$\mathbf{M} = \left[K_0(t)\mathbf{H}_j \; K_1(t)\mathbf{H}_j \; \ldots \; K_{i-1}(t)\mathbf{H}_j \right] \in \mathbb{R}^{i \times \text{clk}}, \tag{5}$$

Fig. 11 demonstrates an example of our methodology. The blue band highlights the LTI identified by the ACA flow method, as detailed in Table 4. Whereas the time-of-interest (TOI) spans clock cycles 0–10 (all cycles in Fig. 11), the ACA flow narrows this down to cycles 3-8 (selected with blue line in Fig. 11). In that example, net (K_0) exhibits the highest architecture correlation with the leakage model $\mathbf{H}_j$. Our approach likewise identifies net0 effectively, even without explicitly defining a precise leakage time interval beforehand. Notably, the cycle-accurate VCD parser step is executed only once, regardless of any subsequent changes to the leakage model $\mathbf{H}_j$. i.e., the larger matrix in this example, which corresponds to the nets' activities over time $\mathbf{K_i(t)}$, remains the same. Only the dot product with different leakage models needs to be recalculated.

Leakage assessment phase - LIF estimation. For each net i, the parser yields a toggle vector; dot products with the HD leakage model produce the architectural correlation C_i. Multiplying C_i by the normalized average gate power P_i/P_T yields the LIF, consistent with the ACA flow. The histograms in Fig. 12 exhibit the typical long-tail shape: in both designs, a single flip-flop dominates the leakage. For the Ibex core, the top-ranked element is a flip-flop within the register file that stores the 128-bit AES state, whereas in the AES coprocessor it is a flip-flop within the 128-bit state register itself. All remaining gates (over 8K nets in Ibex and 37K nets in the AES coprocessor) contribute only marginally.

Timing results (concise). Processing 600 traces on Ibex core completes in **1 h 53 min** (Table 6). A one-off *pre-processing* phase (VMEM generation + gate-level synthesis) accounts for only ≈ 6% (∼ 7 min) of the end-to-end runtime. The VCD dump dominates at ≈ 56%, peak localisation adds ≈ 29%, and all remaining steps together contribute the final ≈ 9%. Overall, the flow achieves a speed-up of **960**× over classical ACA and **1,030**× over PATCH on the same Ibex benchmark.

Processing the AES coprocessor design (1,500 traces) completes in **2 h 42 min**. Although the design is approximately 4.7× larger than Ibex, its shorter 11-cycle operation substantially reduces the time required for generating power traces. About 46% of the total runtime is spent on VCD dumping, while parallelism and the lower cycle count make the VCD parsing phase faster, consuming only ∼ 2% of the total time. Netlist parsing and LIF estimation remain the most time-intensive software phase (∼ 31%), primarily due to the higher gate count. It is also notable that, owing to this larger design, the per-gate power scan with CPT (Joules) takes slightly longer than in Ibex, yet still accounts for only about 2% of the total runtime. Overall, our method achieves a **3.8**× speed-up compared to the ACA flow on this design.

Runtime comparison and methodology summary. In Table 7 we provide a summary of the systematic comparison of three state-of-the-art pre-silicon PSC leakage evaluation methodologies RTL-PAT, PATCH, and ACA, and compare them with the proposed methodology ASSESS. We then list the advantages and drawbacks of each methodology and summarize their runtimes based on our experiments. Like all gate-level methodologies, ASSESS remains dependent on a technology-specific standard-gate library and at least one commercial power estimation run to obtain the LIFs. Data collection cost and runtime for ASSESS are much lower, primarily due to the reduced reliance on time-series power trace. The sampling rate reported in Table 6 is used only for identifying the TOI during peak localization. Once the TOI is determined, the leakage assessment itself is performed using cycle-accurate switching activity, which follows the same mathematical principle as the correlation used in ACA. Since our methodology intentionally considers a broader time window rather than narrowly defined leakage intervals, the risk of missing relevant leakage activity is reduced.

Pre-silicon runtime depends primarily on the number of simulated samples (*latency* × *samples/cycle*) and tool/node settings rather than netlist size alone. As Table 8 shows, ACA's 11-cycle AES coprocessor costs ≈ 18 s per trace, whereas our Ibex running Tiny-AES executes ≈ 10,000 cycles and thus ≈ 2 h 54 m per trace—consistent with this dependency, the total column indicates the approximate

Table 6: ASSESS end-to-end runtime on Ibex and on the AES coprocessor.

Phases	Step	Ibex (600 traces) Per-trace	Ibex (600 traces) Total	AES coprocessor (1500 traces) Per-trace	AES coprocessor (1500 traces) Total
Sim.	Vmem gen. (shared)	141 ms	1 m 24 s	—	—
	Synthesis (once)	—	5 m 53 s	—	36 m 9 s
	VCD dump	6.3 s	1 h 3 m	3 s	1 h 15 m
Pwr. Est.	*Step 1: Peak localisation*	3.3 s	33 m 6 s	—	—
	Step 2: Cycle-accurate VCD parser	56.4 it/s	2 m 28 s	10.4 it/s	2 m 24 s
	Per-gate power scan	—	19 s	—	2 m 49 s
	Calculate C_i	—	3 m 31 s	480 it/s	1 m 18 s
Lkg.A.	Net-list parsing & LIF	40 nets/s	3 m 28 s	12.5 nets/s	50 m 12 s
end-to-end runtime			**1 h 53 m 9 s**		**≈ 2 h 42 m**

Table 7: Comparison of pre-silicon PSC leakage evaluation methodologies.

		ASSESS (this work)	ACA [23]	PATCH [20]	RTL-PAT [16]
Design stage		Gate level	Gate level	Gate level	RTL
Technology dependency		Yes	Yes	Yes	No
CPT dependency		Yes	Yes	Yes	No
Localization granularity		Fine (net/gate)	Fine (net/gate)	Medium (net)	Coarse (module)
Toggle		✔	✔	✔	✔
Delay-aware		✔	✔	✔	✗
Sample rate		2	user defined	user defined	2
Advantage	Leakage assessment at gate level	✔	✔	✔	✗
	LIF per gate	✔	✔	✗	✗
Drawback	Data collection cost	Low	High	High	Medium
	Probability of undiscovered leakage sources	Low	Low	Medium	High
end-to-end* runtime	data collection	≈ 1 hour	> 72 days	> 72 days	≈ 1 hour
	analysis	< 1 hour	> 2 hours	> 6 hours	< 1 hour

* Data from Tables 2, 3, 5, and 6.

time required to collect 600 traces. RTL-PAT [16], with > 3k cycles, reports an end-to-end runtime of ≈ 30 h. Fig. 13 further illustrates this trend: the projected speed-up increases with both design size and the number of simulated cycles. In regions where these factors are high, the proposed method provides the greatest advantage, as power-trace extraction becomes the dominant bottleneck in traditional ACA-style flows.

The cycle-accurate, VCD-driven, gate-level ASSESS methodology achieves the *fine localization granularity* of the classical ACA approach. By parsing toggles once and avoiding repeated CPT trace simulators, ASSESS reduces the end-to-end runtime on Ibex RISC-V core from **72 days to** ≈ 1 **h**. It is roughly 10^3× faster than PATCH and delivers a further 9% improvement over RTL-PAT while offering much finer localization granularity. During analysis, each net's toggles are captured at every clock edge after delay annotation.

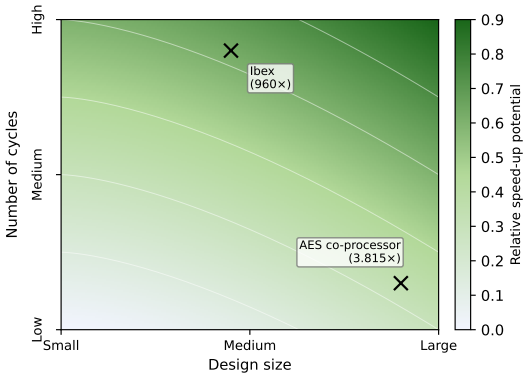


Figure 13: ASSESS speed-up potential.

6 Conclusions

This paper presented a thorough qualitative and quantitative comparison of three prominent pre-silicon power side-channel analysis methodologies, namely RTL-PAT, PATCH, and ACA, and introduced an enhanced ACA-inspired methodology called ASSESS. Our experiments, performed on a RISC-V CPU core running Tiny-AES at the 22nm technology node, revealed the following. RTL-PAT is fast and can be applied early in the design cycle, but lacks gate-level localization. PATCH and ACA offer more fine-grained leakage localization, however, both incur considerable runtime overhead due to extensive time-series power extraction. ASSESS is a cycle-accurate, VCD-driven methodology operating at the gate-level abstraction. It retains ACA's fine-grained localization ability while avoiding its most time-consuming steps. Our experimental results validate that ASSESS substantially accelerates analysis without sacrificing accuracy, making it better suited for iterative ASIC design cycles in advanced technology nodes. By open-sourcing all implementations, we aim to establish a common baseline for future improvements in pre-silicon PSC leakage evaluations. Ultimately, our work highlights the importance of robust pre-silicon PSC leakage evaluations and provides a practical methodology for designing secure hardware in modern ASIC design flows.

Table 8: Complexity of benchmarks & CPT runtimes

	Benchmark		CPT Runtime	
	# gates	# cycles	per trace	total
AES coprocessor [3]	≈38k	11	≈18 s	≈3 h
SoC bus [23]	100k	≈26 (LTI)	≈329 s	≈60 h
Ibex (this work)	≈8k	≈10k	2 h 54 m	≈70 d

Acknowledgments

This work was funded by the Dutch Research Council (NWO) through the PROACT project [1] (NWA.1215.18.014).

References

[1] Asmita Adhikary et al. 2024. PROACT: Physical Attack Resistance of Cryptographic Algorithms and Circuits with Reduced Time to Market. In *ARC'24*. 255–266.

[2] Shivam Bhasin, Jean-Luc Danger, Tarik Graba, Yves Mathieu, Daisuke Fujimoto, and Makoto Nagata. 2013. Physical Security Evaluation at an Early Design-Phase: A Side-Channel Aware Simulation Methodology. In *Proceedings of International Workshop on Engineering Simulations for Cyber-Physical Systems* (Dresden, Germany) *(ES4CPS '14)*. Association for Computing Machinery, New York, NY, USA, 13–20. doi:10.1145/2589650.2559628

[3] ProjectVault Community. 2015. AES Verilog Implementation (Project Vault). https://github.com/ProjectVault/orp/tree/master/hardware/mselSoC/src/systems/geophyte/rtl/verilog/crypto_aes/rtl/verilog. Accessed: 2025-12-12.

[4] Pasquale Davide Schiavone, Francesco Conti, Davide Rossi, Michael Gautschi, Antonio Pullini, Eric Flamand, and Luca Benini. 2017. Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. 1–8. doi:10.1109/PATMOS.2017.8106976

[5] G. Hollestelle, W. Burgers, and Jeremy den Hartog. 2004. *Power analysis on smartcard algorithms using simulation.* Number 22 in CSR-04. Eindhoven University of Technology, Netherlands. Imported from DIES.

[6] Pantea Kiaei, Zhenyuan Liu, Ramazan Kaan Eren, Yuan Yao, and Patrick Schaumont. 2021. Saidoyoki: Evaluating side-channel leakage in pre- and post-silicon setting. Cryptology ePrint Archive, Paper 2021/1235. https://eprint.iacr.org/2021/1235

[7] Pantea Kiaei, Yuan Yao, Zhenyuan Liu, Nicole Fern, Cees-Bart Breunesse, Jasper Van Woudenberg, Kate Gillis, Alex Dich, Peter Grossmann, and Patrick Schaumont. 2024. Gate-Level Side-Channel Leakage Ranking With Architecture Correlation Analysis. *IEEE Transactions on Emerging Topics in Computing* 12, 2 (2024), 496–507. doi:10.1109/TETC.2023.3268303

[8] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Advances in Cryptology — CRYPTO' 99*, Michael Wiener (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 388–397.

[9] kokke. 2024. tiny-AES-c (Small portable AES128/192/256 in C). GitHub repository. Available at: https://github.com/kokke/tiny-AES-c, Accessed: August 14, 2024.

[10] Haocheng Ma, Qizhi Zhang, Ya Gao, Jiaji He, Yiqiang Zhao, and Yier Jin. 2022. PathFinder: side channel protection through automatic leaky paths identification and obfuscation. In *Proceedings of the 59th ACM/IEEE Design Automation Conference* (San Francisco, California) *(DAC '22)*. Association for Computing Machinery, New York, NY, USA, 79–84. doi:10.1145/3489517.3530413

[11] Stefan Mangard, Maria Elisabeth Oswald, and Thomas Popp. 2007. *Power Analysis Attacks - Revealing the Secrets of Smart Cards.* Springer. XXIII, 337 S..

[12] Stefan Mangard and Kai Schramm. 2006. Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, Louis Goubin and Mitsuru Matsui (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 76–90.

[13] Daniel Moghimi, Berk Sunar, Thomas Eisenbarth, and Nadia Heninger. 2020. TPM-Fail: TPM meets timing and lattice attacks. In *Proceedings of the 29th USENIX Conference on Security Symposium (SEC'20)*. USENIX Association, USA, Article 116, 17 pages.

[14] Colin O'flynn and Zhizhang Chen. 2014. Chipwhisperer: An open-source platform for hardware embedded security research. In *Constructive Side-Channel Analysis and Secure Design: 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers 5*. Springer, 243–260.

[15] Vamshi PN. [n. d.]. pnvamshi/Hardware-Implementation-of-AES-Verilog: Hardware Implementation of Advanced Encryption Standard Algorithm in Verilog. https://github.com/pnvamshi/Hardware-Implementation-of-AES-Verilog

[16] Nitin Pundir, Jungmin Park, Farimah Farahmandi, and Mark Tehranipoor. 2022. Power Side-Channel Leakage Assessment Framework at Register-Transfer Level. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 30, 9 (2022), 1207–1218. doi:10.1109/TVLSI.2022.3175067

[17] Francesco Regazzoni, Stephane Badel, Thomas Eisenbarth, Johann Grobschadl, Axel Poschmann, Zeynep Toprak, Marco Macchetti, Laura Pozzi, Christof Paar, Yusuf Leblebici, and Paolo Ienne. 2007. A Simulation-Based Methodology for Evaluating the DPA-Resistance of Cryptographic Functional Units with Application to CMOS and MCML Technologies. In *2007 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*. 209–214. doi:10.1109/ICSAMOS.2007.4285753

[18] Rajat Sadhukhan, Paulson Mathew, Debapriya Basu Roy, and Debdeep Mukhopadhyay. 2019. Count Your Toggles: a New Leakage Model for Pre-Silicon Power Analysis of Crypto Designs. *J. Electron. Test.* 35, 5 (Oct. 2019), 605–619. doi:10.1007/s10836-019-05826-8

[19] Abolfazl Sajadi, Nusa Zidaric, Todor Stefanov, and Nele Mentens. 2024. A Systematic Comparison of Side-channel Countermeasures for RISC-V-based SoCs. In *2024 IEEE Nordic Circuits and Systems Conference (NorCAS)*. 1–7. doi:10.1109/NorCAS64408.2024.10752477

[20] Vahhab Samadi Bokharaie and Ali Jahanian. 2022. Power side-channel leakage assessment and locating the exact sources of leakage at the early stages of ASIC design process. *J. Supercomput.* 78, 2 (Feb. 2022), 2219–2244. doi:10.1007/s11227-021-03927-w

[21] Patanjali Slpsk, Prasanna Karthik Vairam, Chester Rebeiro, and V. Kamakoti. 2019. Karna: A Gate-Sizing based Security Aware EDA Flow for Improved Power Side-Channel Attack Protection. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8. doi:10.1109/ICCAD45719.2019.8942173

[22] Amisha Srivastava, Sanjay Das, Navnil Choudhury, Rafail Psiakis, Pedro Henrique Silva, Debjit Pal, and Kanad Basu. 2024. SCAR: Power Side-Channel Analysis at RTL Level. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 32, 6 (2024), 1110–1123. doi:10.1109/TVLSI.2024.3390601

[23] Yuan Yao, Tarun Kathuria, Baris Ege, and Patrick Schaumont. 2020. Architecture Correlation Analysis (ACA): Identifying the Source of Side-channel Leakage at Gate-level. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 188–196. doi:10.1109/HOST45689.2020.9300271

[24] Tao Zhang, Jungmin Park, Mark Tehranipoor, and Farimah Farahmandi. 2021. PSC-TG: RTL Power Side-Channel Assessment with Test Pattern Generation. doi:10.1109/DAC18074.2021.9586210