ZK-Audit: Proving Power Side-Channel Resilience in Synthesized Hardware

Sakib Anwar Rieyan and Nektarios Georgios Tsoutsos

University of Delaware
{rieyan, tsoutsos}@udel.edu

Abstract. Modern hardware security heavily relies on the assumption that pre-synthesis algorithmic protections will survive the physical fabrication pipeline. However, untrusted third-party Electronic Design Automation (EDA) toolchains often apply aggressive structural optimizations that can silently compromise perfectly symmetric designs, introducing critical data-dependent power side-channel vulnerabilities. Existing pre-silicon verification methodologies require exposing highly sensitive, proprietary gate-level intellectual property (IP) to external auditors to verify structural security. In this paper, we introduce a novel Zero-Knowledge Hardware Auditor, an end-to-end framework that provides mathematical guarantees of physical data-obliviousness without revealing the underlying circuit netlist. By translating synthesized gate-level topologies into a custom Side-Channel Intermediate Language (SCIL), our architecture maps physical dynamic switching activity into arithmetic constraints executable within a Halo2 zero-knowledge virtual machine (zkVM). This enables the first implementation of a zero-knowledge Bounded Toggle Assessment (ZK-BTA), a deterministic structural counterpart to classical Test Vector Leakage Assessment. Experimental evaluations across standard cryptographic primitives and ISCAS-85 benchmarks demonstrate that the framework successfully identifies inherently leaky logic and captures EDA-induced asymmetries, such as a 13% leakage rate introduced into a theoretically secure Montgomery Ladder, while proving the structural integrity of Dual-Rail oblivious logical topologies. Furthermore, the asymmetric zk-SNARK architecture ensures scalable component-level auditing, yielding a succinct cryptographic proof of physical security that can be publicly verified in under 0.08 seconds.

Keywords: Hardware Security · Zero-Knowledge Proofs · Side-Channel Analysis · Pre-Silicon Verification · Electronic Design Automation · zk-SNARKs · Data-Oblivious Circuits.

1 Introduction

Although modern cryptographic primitives offer robust mathematical guarantees of confidentiality and integrity, the physical hardware structures that execute these algorithms inevitably leak sensitive information through auxiliary channels. Power side-channel attacks (SCAs) exploit data-dependent dynamic power

consumption arising from the charging and discharging of capacitive nodes during active logic gate switching [15]. Ever since the foundational introduction of Differential Power Analysis (DPA), it has been well-established that an adversary can correlate physical power measurements with hypothetical key-dependent intermediate states to systematically extract full cryptographic keys without undermining the mathematical security of the underlying cipher [15].

In contemporary computing environments, the threat landscape has expanded significantly beyond isolated cryptographic modules. Modern hardware designs incorporate intricate Systems-on-Chip (SoC) architectures, where shared memory buses, complex interconnects, and heterogeneous compute fabrics introduce hidden data-dependent leakage paths that compromise otherwise secure cores [13]. This vulnerability extends directly to lightweight cryptographic candidates evaluated for hardware deployment, where design constraints often restrict the implementation of comprehensive physical countermeasures [18]. Furthermore, side-channel vulnerabilities are no longer limited to microelectronic hardware with physical proximity requirements; they actively threaten virtualized cloud architectures and hardware-backed confidential virtual machines through microarchitectural tracking [29]. Traditional timing-based leakages [16] have similarly evolved, with modern microarchitectural features like Dynamic Voltage and Frequency Scaling (DVFS) enabling attackers to turn local power fluctuations into remote, network-exploitable timing channels [26]. As a result, early-stage pre-silicon side-channel assessment has evolved from an isolated design consideration into a sign-off requirement for secure hardware validation.

Achieving secure hardware realization is fundamentally constrained by a critical trade-off between detection accuracy, execution complexity, and mitigation costs across the Electronic Design Automation (EDA) abstraction flow. Post-silicon verification provides absolute physical fidelity by capturing true parasitic effects, substrate coupling, and process variations, yet it occurs too late in the product development lifecycle to permit cost-effective structural remediation, turning a minor logic fix into a multi-million dollar tapeout respin [28]. To eliminate vulnerabilities early, behavioral screening frameworks like RTL-PSC operate at the Register-Transfer Level (RTL) to evaluate toggle-activity distributions using information-theoretic metrics like Kullback-Leibler (KL) divergence [12]. However, behavioral abstractions are inherently limited because they fail to capture physical synthesis artifacts, layout capacitances, and critical timing hazards [21].

To bridge this gap, electronic design workflows rely heavily on gate-level netlist assessments. Frame-based simulation frameworks, such as Architecture Correlation Analysis (ACA) and its fine-grained gate-level ranking extensions, help isolate the exact percentage of logic cells responsible for localized data leakage [14, 27]. Similarly, structural testing frameworks incorporate statistical Test Vector Leakage Assessment (TVLA) methodologies directly into hardware verification environments to evaluate design-time sign-off criteria [1]. To bypass the massive computational overhead of simulating millions of toggle vectors, analytical netlist methodologies like FORTIFY build directed circuit graphs to

evaluate a signal's security using static Signal Probability Correlation Factors (SPCF) [17]. Machine learning models and optimization heuristics, such as the Monte Carlo Tree Search (MCTS) utilized in the ASCENT framework, have also been introduced to rapidly navigate the expansive logic synthesis design space and predict side-channel resilience [4], while Generative Adversarial Networks (GANs) are leveraged to accelerate complex electromagnetic layout evaluations [10].

Despite these advancements, conventional gate-level netlist auditing remains heavily constrained by the optimization objectives of automated logic synthesis engines. Standard EDA compilers optimize aggressively for Power, Performance, and Area (PPA), frequently destroying the structural balance of data-oblivious dual-rail logic or introducing physical glitches: transient switching activities that occur before logic nodes settle to their final values [6]. These glitches can momentarily recombine randomized secret shares within masked implementations, entirely invalidating theoretical security guarantees [6]. While formal verification tools can verify non-interference properties to mathematically prove algorithmic side-channel security [2, 6], they are highly susceptible to state-space explosion when applied to deep, highly non-linear cryptographic components.

To address the limitations of computationally intensive trace simulations and vulnerable heuristic abstractions, this paper introduces a novel *Zero-Knowledge Power Auditing Framework* for synthesizable hardware netlists. Rather than attempting to model physical waveforms or rely on heuristic power profiles, our approach encodes the gate-level netlist directly into a verifiable mathematical constraint system, leveraging a Zero-Knowledge Virtual Machine (zkVM) to verify data-oblivious execution.

The primary contributions of this work are outlined as follows:

- **SCIL Intermediate Representation (IR).** We present an automated hardware compilation pipeline that translates standard structural netlists into a universally verifiable arithmetic constraint format, decoupling the security evaluation from proprietary EDA tools.
- **Deterministic Verification of EDA-Induced Asymmetry.** Unlike traditional simulation-based TVLA frameworks that rely on physical noise models and empirical statistical thresholds, our zkVM deterministically evaluates the unrolled gate-level execution paths. This permits the framework to deterministically identify asymmetric gate routing or unbalanced load distributions introduced during cell mapping and optimization.
- **zkVM for Provable Bounded Leakage.** We introduce a randomized, zero-knowledge variant of the Bounded Toggle Assessment methodology. By executing a Monte Carlo input generation routine that tests extreme data transitions over the compiled arithmetic circuit, our framework calculates a precise Leakage Detection Rate (LDR). This metric allows designers to mathematically demonstrate an exact upper bound for physical data dependency (ϵ-bounded leakage) prior to physical layout execution.

To establish a clear verification boundary, our framework evaluates hardware side-channel resistance strictly at the pre-silicon structural netlist level under

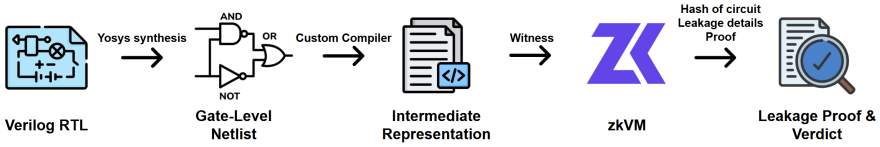


Fig. 1. High-level architectural overview of the privacy-preserving zero-knowledge hardware auditing pipeline.

a steady-state power consumption model. The cryptographic proof guarantees generated by our zkVM are bounded by structural data-obliviousness, tracking deterministic gate transitions exclusively at stable clock cycle boundaries.
Roadmap: The remainder of this paper is organized as follows. Section 2 discusses background and our threat model. Section 3 describes the proposed framework, while Section 4 introduces our methodology. Section 5 evaluates our framework on cryptographic benchmarks, while Section 6 discusses comparisons with prior works. Finally, Section 7 concludes the paper and outlines future work.

2 Background and Threat Model

2.1 Differential Power Analysis (DPA)

In standard Complementary Metal-Oxide-Semiconductor (CMOS) technology, the primary source of data-dependent power dissipation is dynamic power, which occurs during the charging and discharging of parasitic load capacitances when logic gates transition between binary states [19]. Because these electrical transitions are dictated by the data being processed, the instantaneous power dissipation of a device correlates directly with its internal computational state. Differential Power Analysis (DPA) exploits this physical vulnerability by utilizing statistical techniques to identify minute variations in power consumption across numerous executions of a cryptographic algorithm [15].

To quantify this data-dependency mathematically, Correlation Power Analysis (CPA) typically employs the Hamming weight model (counting the number of high bits in a data word) or the Hamming distance model (measuring the number of bit-flips between consecutive register states) to approximate the dynamic power signature [8]. By statistically correlating these hypothetical leakage models with physical oscilloscope measurements, an attacker can isolate the correct secret key from the background electrical noise, entirely bypassing the algorithmic security of the underlying cipher [15, 19].

2.2 Zero-Knowledge Proofs in Hardware

Zero-Knowledge Proofs (ZKPs) are cryptographic protocols that enable a prover to mathematically demonstrate the validity of a computation to a verifier with-

out revealing the underlying inputs [23]. While historically applied to privacy-preserving software and blockchain scalability, the foundational mechanics of ZKPs translate computational logic into verifiable arithmetic circuits, mapping seamlessly to hardware security validation.

Within a Zero-Knowledge Virtual Machine (zkVM), a synthesized gate-level netlist can be deterministically unrolled and encoded as a strict constraint system. Modern arithmetization schemes, such as those utilized in PLONK and its derivatives, allow standard Boolean logic gates (for instance, AND, XOR, NOT) and complex routing multiplexers to be transformed into high-degree polynomial identities and permutation arguments [9]. To implement this, our framework leverages Halo2, an advanced zk-SNARK proving system built upon PLONK-ish arithmetization. Halo2 is particularly advantageous for hardware modeling because it supports highly flexible custom gates and lookup tables. This allows standard logic operations (for instance, AND, XOR, NOT) and complex routing multiplexers to be densely packed into an execution matrix rather than relying on inefficient traditional constraint boundaries.

By committing to the execution trace of the hardware circuit using polynomial commitments, the zkVM can verify the structural switching activity of the netlist across varying inputs. This allows the framework to mathematically prove that the switching activity (and thus, the dynamic power footprint) remains perfectly symmetrical or bounded. This effectively advances traditional formal hardware verification techniques [2, 6] by addressing physical power asymmetries without triggering the state-space explosion typically associated with deep sequential circuits.

2.3 Threat Model

We assume a standard, passive side-channel adversary equipped with physical access to the target cryptographic device [19]. The attacker possesses the capability to record high-resolution traces of the device's dynamic power consumption while it processes known or chosen data inputs (for example, plaintexts or ciphertexts). The adversary's primary objective is to recover the embedded cryptographic secret key by correlating data-dependent power leakages [15].

In addition to the adversarial setting, this work is motivated by a practical deployment scenario in which a hardware IP vendor wishes to provide a cryptographic core to a client without revealing proprietary implementation details. In such an environment, traditional verification is not sufficient because the client cannot directly inspect the source code or internal design to assess whether the implementation is resistant to side-channel leakage. This creates a trust gap between the vendor and the verifier.

To address this, we consider a ZKP-based verification setting where the vendor can mathematically prove that the implemented design satisfies a formally defined leakage bound without disclosing the underlying netlist or source code. Unlike conventional debugging or self-verification, where a designer can trivially inspect and test their own implementation, the ZKP framework is necessary specifically because the verifier does not have access to the design internals and

must still be convinced of its security properties. For the pre-silicon verification framework proposed in this paper, we assume the vendor's automated toolchain has full access to the synthesized gate-level netlist. We further assume that underlying EDA synthesis tools (e.g., logic optimizers and technology mappers) are untrusted and may introduce unintended asymmetries or glitch-inducing paths that violate data-oblivious design principles [21].

Active physical tampering, such as hardware Fault Injection (FI) attacks [7], as well as remote microarchitectural timing attacks [16], are considered out of scope for this power-auditing methodology. Our framework strictly limits its guarantees to proving that instantaneous dynamic power variations of the synthesized netlist remain below a mathematically defined noise threshold (ϵ) across the evaluated bounded test vectors. For our evaluations, ϵ is heuristically set to a 5% maximum allowable toggle discrepancy, though strict cryptographic deployments may enforce tighter bounds.

3 The Zero-Knowledge Hardware Auditor Architecture

3.1 Framework Overview

The proposed zero-knowledge hardware auditor architecture provides a completely non-interactive, privacy-preserving pipeline to verify the data-obliviousness of a hardware circuit prior to fabrication. As illustrated by the high-level system overview in Fig. 1 and detailed in the granular architectural workflow in Fig. 2, the pipeline structurally detaches the untrusted manufacturing and synthesis toolchain from the cryptographic verification environment. Overall, the end-to-end flow operates across four tightly coupled stages:

1. **Stage 1: Synthesis (Untrusted Domain):** The architecture takes a designer's high-level hardware description language specification (*Verilog RTL*) as its initial input. This RTL description is processed by an untrusted synthesis framework, such as the open-source *Yosys Synthesis* suite, to perform technology mapping down to a structural *Gate-Level Netlist*. As highlighted in Stage 1 of Fig. 2, this stage represents a critical security boundary: optimizations or cell selections performed here are completely untrusted and prone to introducing structural or timing asymmetries.
2. **Stage 2: Translation (Intermediate Domain):** The untrusted gate-level netlist is ingested by our custom compiler that converts the physical wire connections and Boolean gates into a domain-specific intermediate representation (IR) designated as *SCIL Constraints* (Side-Channel Intermediate Language). This stage flattens the logical topology into a set of discrete arithmetic relations over a finite field (for instance, mapping operations to structured rows such as `NAND 1 2 3 4 10` or `XOR 4 5 6 15`, where the numerical sequence denotes the unique labels of the connected wires followed by the gate's derived parasitic weight).
3. **Stage 3: Zero-Knowledge Proving (Secure Domain):** The SCIL constraints are passed to the core execution environment: the *Halo2 zkVM*

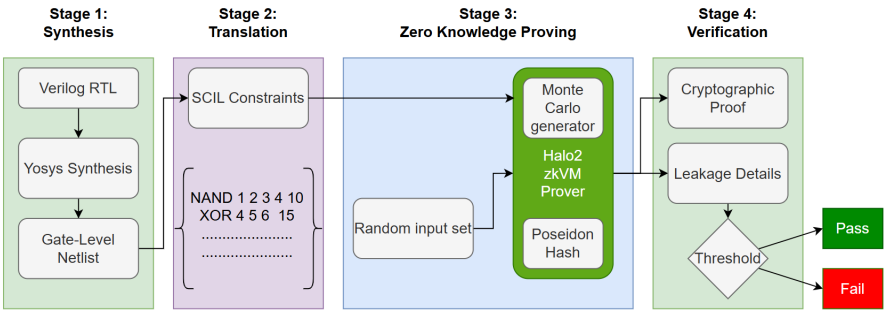


Fig. 2. Detailed step-by-step cryptographic translation and verification workflow across the four pipeline stages.

Prover. Concurrently, a *Monte Carlo generator* creates randomized input vectors. The prover executes the circuit logic using these test sets, generating a deterministic execution trace. To guarantee that a malicious prover cannot alter the underlying netlist between Monte Carlo iterations, our framework computes a cryptographically secure *Poseidon Hash* over the compiled SCIL circuit structure, ensuring the exact same hardware logic is evaluated across all randomized inputs. The prover outputs a zero-knowledge succinct non-interactive argument of knowledge (*Cryptographic Proof*) alongside bounded side-channel *Leakage Details*.

4. **Stage 4: Verification (Public Domain):** In the final verification block, the public auditor evaluates the cryptographic proof for mathematical validity. Simultaneously, a hardware *Comparator* tests the verified leakage metrics against a user-defined threshold (ϵ). If the proof is valid and the structural data dependency falls below the designated threshold boundary, the system returns a **Pass** status, indicating a provably data-oblivious logical topology. Otherwise, a **Fail** status is emitted, revealing the presence of an exploitable leakage vulnerability.

3.2 The SCIL Intermediate Representation

A fundamental challenge in auditing physical side-channel behavior inside a zero-knowledge virtual machine is the vast conceptual mismatch between digital switching hardware and polynomial equations over finite fields. To reconcile this, we introduce the Side-Channel Intermediate Language (SCIL). As shown in the Stage 2 translation block of Fig. 2, the primary purpose of the SCIL IR compiler is to convert structural gate-level netlists into a system of mathematically provable arithmetic constraints without losing topological or temporal tracking properties.

Every standard-cell gate parsed from the netlist is mapped onto an algebraic constraint equation within the constraint system. For a standard two-input

Boolean NAND gate with inputs $a, b \in \{0, 1\}$ and output $c \in \{0, 1\}$, the SCIL compiler enforces the strict arithmetization layout:

$$a \cdot b + c - 1 = 0.$$

Similarly, for a two-input XOR gate with inputs a, b and output c, the polynomial constraint is generated as:

$$a + b - 2ab - c = 0.$$

By serializing the entire netlist into these algebraic primitives, SCIL creates a strict, index-mapped sequence of statements. Because the intermediate language preserves the exact connection graph of the netlist, any physical toggle occurring on an internal wire corresponds deterministically to a specific constraint witness evaluation row within the zkVM matrix. This allows the compiler to securely abstract away raw structural hardware details into a uniform algebraic format ready for parallel execution tracing.

3.3 The ZK-BTA Verification Engine

To establish empirical side-channel guarantees within a zero-knowledge context, our framework implements a Bounded Toggle Assessment (BTA), natively executed inside the zkVM circuit boundary (visualized in Stage 3 of Fig. 2). Rather than relying on physical, noisy oscilloscope measurements taken from real silicon, the ZK-BTA verification engine evaluates deterministic structural leakage over simulated execution traces.

Our framework's Monte Carlo generator feeds two separate input classes: Bucket A (for example, fixed or extreme input conditions) and Bucket B (for example, fully randomized data inputs); into the arithmetized SCIL constraint logic. As the Halo2 prover constructs the trace polynomial commitments for both buckets, it calculates the cumulative internal switching activity across all internal nodes.

The core proof obligation of the ZK-BTA engine is to mathematically demonstrate that the difference in dynamic switching activity between Bucket A and Bucket B stays within a strict, predefined noise margin. Let Δ_A and Δ_B represent the total calculated gate-toggle transitions for input sets A and B, respectively. The hardware comparator evaluates the absolute delta relation:

$$|\Delta_A - \Delta_B| < \epsilon.$$

By integrating this comparison box directly into the zk-SNARK constraint pipeline, the engine provides an unprecedented structural guarantee: it can cryptographically prove to an external party that a netlist's power signature is mathematically bounded and data-oblivious, without ever revealing the proprietary intellectual property (IP) of the circuit netlist, the underlying test vectors, or the raw execution traces.

4 Our Methodology

To accurately model physical side-channel power leakage prior to silicon tape-out, our methodology uses gate-level switching activity as a direct, programmatically trackable proxy for dynamic power consumption (P_{dyn}). By abstracting the hardware into the Side-Channel Intermediate Language (SCIL), our framework's core algorithms can iterate over any generalized gate-level netlist and deterministically map state transitions into the zkVM witness matrix. Following standard conventions in pre-silicon side-channel evaluation, the current implementation models switching activity using a symmetric Hamming Distance approximation, assigning equal leakage contribution to rising and falling transitions.

SCIL uses gate-level parasitic capacitance values derived from the target standard-cell library. In this work, we use the FreePDK45 [22] standard cell library, where each logic gate is associated with a pre-characterized input capacitance. These capacitance values are used as weights in the SCIL intermediate representation during cost modeling. Thus, gate "weights" are not assumed uniform but are directly extracted from the synthesis library.

4.1 Dynamic Power Modeling via State Transitions

In standard CMOS digital logic circuits, dynamic power dissipation is dominated by the charging and discharging of internal node capacitances during logic state changes. We formally model this relationship for an arbitrary circuit transitioning from an initial temporal state vector (V_{t0}) to a subsequent state vector (V_{t1}) as:

$$P_{dyn} = \sum_{i=1}^{N} SA_i \cdot \alpha C_i,$$

where N represents the total number of gates in the compiled netlist, αC_i represents the physical parasitic capacitance weight of gate i, and SA_i represents the discrete Switching Activity coefficient.

Within our zkVM implementation, the switching activity parameter is defined via a logical exclusive-OR operation across the temporal boundaries. The prover codebase calculates this dynamically for every node i:

$$SA_i = V_{t0,i} \oplus V_{t1,i} \quad \text{where} \quad SA_i \in \{0, 1\}.$$

4.2 zkVM Witness Generation and Audit Logging

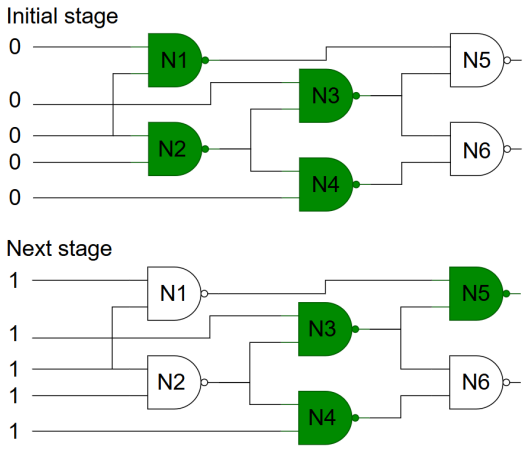
The core methodology relies on translating these temporal transitions into verifiable cryptographic constraints. For any target circuit, the compiler instantiates a witness matrix where each column tracks the temporal state of a specific internal wire.

As the Halo2 prover executes the randomized input vectors, it records the initial state V_{t0} and the next state V_{t1} in adjacent execution rows. The framework

natively compiles an automated "audit log", a mathematical aggregation of all SA_i values across the constraint system. Because this logic is generalized, the framework can parse circuits containing thousands of gates, programmatically identifying which nodes toggle ($SA = 1$) and which remain static ($SA = 0$) without requiring physical simulation.

4.3 Illustrative Execution: The c17 Benchmark

To demonstrate how the generalized framework processes state transitions and generates this audit log, we examine an execution trace of the standard ISCAS-85 `c17` benchmark. Figure 3 illustrates the framework's evaluation of an all-NAND topology under an extreme input vector swap, transitioning uniformly from an initial state of `0` to a subsequent state of `1` ($V_{t0} : \{0, 0, 0, 0, 0\} \rightarrow V_{t1} : \{1, 1, 1, 1, 1\}$).



Gate	Init	Next	Toggled?	Power
N1	1	0	1	1×αC
N2	1	0	1	1×αC
N3	1	1	0	0
N4	1	1	0	0
N5	0	1	1	1×αC
N6	0	0	0	0
			∑SA=3	3αC

Fig. 3. Gate-level switching activity analysis and embedded audit log execution for the ISCAS-85 `c17` benchmark under a global $0 \rightarrow 1$ transition.

As documented in the embedded audit log table within Fig. 3, the zkVM logic traces the cascading switching effect programmatically. When the primary inputs step high, the internal nodes adjust according to the strict SCIL constraints. The framework detects that gates $N1$, $N2$, and $N5$ undergo state flips, registering individual SA values of `1`. Gates $N3$, $N4$, and $N6$ maintain static logic states across the cycle boundary, recording SA values of `0`.

For this specific transition, the prover aggregates the individual gate states to compute the localized side-channel footprint:

$$\sum SA = 3 \quad \Longrightarrow \quad \sum P_{dyn} = 3\alpha C.$$

4.4 Global Leakage Assessment

While the `c17` example demonstrates a single transition, the overarching methodology applies this exact computation in parallel across millions of randomized Bounded Toggle Assessment (BTA) inputs. By relying on the deterministic $\sum SA$ calculated by the codebase, our framework establishes a verifiable mathematical baseline. It then calculates the absolute difference between input distributions (for example, Bucket A vs. Bucket B). If this delta exceeds the allowable threshold (ϵ), the framework algorithmically uncovers structural asymmetry boundaries without risking proprietary netlist leakage.

5 Evaluation and Experimental Results

To assess the practical viability and security guarantees of the proposed Zero-Knowledge Hardware Auditor, we evaluated our framework across a suite of standard cryptographic primitives and ISCAS-85 benchmark circuits. The evaluation focuses on two primary metrics: the system's ability to accurately detect data-dependent side-channel leakage post-synthesis, and the computational overhead of the zkVM execution trace.

5.1 Security Analysis and Leakage Detection

The framework's verification engine was subjected to a simulated Bounded Toggle Assessment (BTA) utilizing 100 randomized Monte Carlo iterations per circuit. The system calculates a Leakage Detection Rate (LDR), representing the percentage of test iterations where the dynamic switching activity delta (ΔSA) exceeded the safe threshold. Standard TVLA requires the number of tests to be much higher to overcome Gaussian noise and measurement artifacts in physical oscilloscopes. Because our zkVM framework relies on noise-free, deterministic algebraic simulation, the Signal-to-Noise Ratio (SNR) is effectively infinite. Therefore, 100 iterations are mathematically sufficient to detect structural asymmetry without needing to brute-force statistical noise. Before every test, the input Buckets are re-initialized and the system checks pass/fail for each of the iterations.

Table 1. Leakage Detection Rate (LDR) across benchmark circuits under 100 BTA iterations.

Circuit	Architecture / Type	LDR	Status
`adder`	Standard CMOS	54%	×
`secure_adder`	Dual-Rail Oblivious	0%	✓
`vuln_modexp`	Sq-and-Multiply	84%	×
`secure_modexp`	Montgomery Ladder	13%	△
`c17`	ISCAS-85 (Basic Logic)	73%	×
`c432`	ISCAS-85 (Priority Decoder)	27%	△
`c499`	ISCAS-85 (Error Correction)	88%	×
`c880`	ISCAS-85 (ALU/Multiplier)	92%	×

✓ **Pass:** Provably secure; oblivious logical topology successfully preserved by EDA tool.
△ **Moderate Leakage:** Expected data dependency or asymmetry introduced via EDA logic optimization.
× **Fail:** Critical security vulnerability; severe side-channel leakage profile detected.

Our analysis of unprotected baseline circuits yielded expected leakage behaviors. The standard CMOS `adder` recorded a 54% LDR, indicating standard data-dependent power leakage. Similarly, the `vuln_modexp` circuit, utilizing a vulnerable square-and-multiply architecture, triggered an 84% LDR, identifying a critical side-channel vulnerability. General-purpose ISCAS-85 circuits also exhibited high baseline leakage; for example, the simple paths in `c17` resulted in a 73% LDR, while the highly non-linear error-correction logic of `c499` resulted in an 88% LDR due to massive Hamming weight differences. The `c880` ALU/multiplier exhibited the highest baseline correlation, with roughly a 92% LDR.

The critical value of the zero-knowledge auditor is demonstrated when evaluating theoretically "secure" cryptographic implementations. When synthesizing a Dual-Rail Oblivious `secure_adder`, the framework recorded a 0% LDR across all 100 iterations. This mathematically proves that the synthesis tool preserved the oblivious logic layout, resulting in a provably secure netlist.

However, the evaluation of the `secure_modexp` circuit (implemented via a Montgomery Ladder) yielded a 13% LDR. While the Montgomery Ladder is algorithmically completely symmetrical and should yield a 0% LDR, the untrusted EDA toolchain introduced underlying structural optimizations during technology mapping that broke this perfect symmetry. The zkVM successfully caught this EDA-induced asymmetry, proving that pre-synthesis algorithmic security does not guarantee post-synthesis physical security.

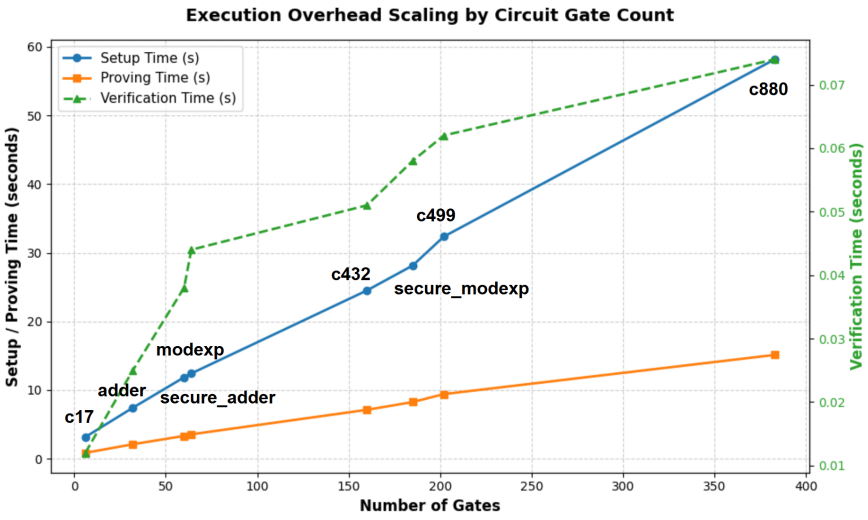


Fig. 4. Execution overhead scaling of the zkVM across Setup, Proving, and Verification phases relative to circuit gate count.

5.2 zkVM Performance and Scalability

A fundamental requirement for any cryptographic auditing tool is computational scalability. We benchmarked the execution overhead of the Halo2 zkVM across three phases: Setup Time, Proving Time, and Verification Time.

As illustrated by the execution overhead analysis in Fig. 4, both Setup Time and Proving Time scale log-linearly with the total number of logic gates in the synthesized netlist. For small-scale logic such as the `c17` benchmark, setup requires approximately 3 seconds, with proof generation taking ∼1 second. As circuit complexity increases to the ∼380-gate `c880` multiplier, Setup Time scales to approximately 58 seconds, and Proving Time scales to roughly 15 seconds.

Crucially, the architecture benefits from the asymmetric nature of zk-SNARKs. While proof generation requires moderate localized compute, the Verification Time remains exceptionally lightweight and detached from the circuit's overall depth. Across all evaluated circuits, verification remained under 0.08 seconds. Verification for the `c17` circuit completed in ∼0.012 seconds, gently scaling to only ∼0.075 seconds for the `c880` benchmark. This asymmetrical scaling ensures that the final mathematical proof of hardware security can be verified almost instantly by external auditors or thin clients without requiring intensive computational resources.

6 Related Work

The proposed Zero-Knowledge Hardware Auditor architecture bridges the gap between formal side-channel verification and cryptographic proof systems. Our framework pioneers the application of zk-SNARKs (via a Halo2 zkVM) to obliviously audit EDA synthesis behavior, extending foundational concepts from established hardware security architectures. We compare our approach against three primary paradigms in the next sections.

6.1 Formal Verification and Leakage Contracts

Traditional pre-silicon power side-channel analysis (PSCA) relies heavily on formal satisfiability (SAT/SMT) solvers to search for gate assignments that violate security properties [5]. Recent advancements, such as the hardware leakage contracts proposed by Haring et al. [11], successfully formalize the boundary between gate-level glitch transitions and masked software. Furthermore, emerging frameworks like Q-LEAK (2026) attempt to solve the severe state-space explosion inherent to SAT-based side-channel verification by utilizing quantum-search algorithms [20].

Comparison: While these modern formal methods guarantee structural security, they suffer from two fundamental limitations. First, traditional SAT/SMT still struggles to scale natively on deeply non-linear components (like multipliers or Montgomery Ladders) without hybrid or quantum assistance. Second, the resulting proofs are strictly private; an external auditor must possess both the proprietary gate-level netlist and the computational environment to re-verify the leakage contract. Our architecture translates this constraint-solving mechanism into a zero-knowledge execution trace, yielding a succinct, bounded-time proof that *publicly* verifies the lack of leakage without ever exposing the proprietary hardware IP to the auditor.

6.2 Pre-Silicon Simulation and Information Flow Tracking

Architectures utilizing Gate-Level Information Flow Tracking (GLIFT) precisely monitor how untrusted inputs influence output behaviors [24]. More recently, modern pre-silicon frameworks [3] rely on extracting heavily simulated execution traces to perform Test Vector Leakage Assessments (TVLA) on AI/ML and cryptographic architectures long before physical tape-out.

Comparison: GLIFT strictly tracks taint flows but fails to model the aggregate dynamic switching activity (P_{dyn}) required to detect physical power correlation, and often requires invasive modifications to the core EDA pipeline. Conversely, modern pre-silicon simulation frameworks successfully model power, but rely on localized, noisy simulation trace generation. By abstracting the netlist into SCIL constraints processed by an isolated zkVM, our architecture provides the gate-level topological precision of GLIFT and the power-modeling capabilities of trace simulation, all while entirely decoupling the security audit from the untrusted synthesis toolchain (e.g., Yosys).

6.3 Verifiable Hardware and Zero-Knowledge Execution

The use of zk-SNARKs in hardware has traditionally focused on "Verifiable Computation" (for instance, ZKML), where a prover demonstrates that a specific hardware accelerator or cloud machine executed an algorithm correctly. Recent works continue to explore physically secure Trusted Execution Environments (TEEs) and zero-knowledge tunneling architectures to protect algorithmic IP and network states [25].

Comparison: Existing ZK hardware architectures focus strictly on *functional* correctness. Our framework uniquely repurposes the zkVM execution matrix to verify *physical and structural* correctness. Instead of merely proving that an algorithm executed correctly, our architecture proves that the underlying synthesized hardware netlist executed with a bounded, mathematically verified data-oblivious switching signature.

7 Conclusion

In this paper, we presented a novel, end-to-end Zero-Knowledge Hardware Auditor designed to mathematically guarantee the physical data-obliviousness of digital circuits prior to silicon fabrication. A fundamental vulnerability in modern hardware design is the implicit trust placed in third-party EDA toolchains.

To address this challenge, our framework structurally detaches the untrusted synthesis pipeline from the security verification process. By translating gate-level netlists into the Side-Channel Intermediate Language (SCIL), we successfully mapped physical hardware topologies into arithmetic constraints executable within a Halo2 zkVM. This enables the successful execution of a zero-knowledge Bounded Toggle Assessment (ZK-BTA) to formally verify data-obliviousness. Our experimental results across standard ISCAS-85 benchmarks and cryptographic primitives show that our framework can accurately distinguish between inherently leaky architectures, EDA-compromised logic, and provably secure layouts. Crucially, the architecture achieves this without exposing the proprietary gate-level IP or the raw execution traces to the external auditor.

Future work will focus on expanding the SCIL compiler to apply asymmetric weighting to rising versus falling transitions, natively model physical gate glitches and routing delays, allowing the zkVM to audit timing side-channels alongside dynamic power consumption. Ultimately, this zero-knowledge architecture represents a significant step toward verifiable, trustless hardware manufacturing, providing external entities with mathematical proof of physical security without compromising intellectual property.

Acknowledgment

The authors would like to acknowledge the support of the National Science Foundation (Award 2453861).

References

1. Khaled Alatoun and Ranga Vemuri. Power side-channel verification in hardware designs. In *Proceedings of NAECON 2024 – IEEE National Aerospace and Electronics Conference*, pages 291–296, Dayton, OH, USA, 2024.
2. José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, and François Dupressoir. Verifiable side-channel security of cryptographic implementations: Constant-time MEE-CBC. In *Fast Software Encryption – FSE 2016*, volume 9783 of *Lecture Notes in Computer Science*, pages 163–184. Springer, 2016.
3. Furkan Aydin, Emre Karabulut, and Aydin Aysu. Pre-silicon side-channel analysis of ai/ml systems. In *2024 IEEE Physical Assurance and Inspection of Electronics (PAINE)*. IEEE, 2024.
4. Jaydeep Bhandari, Animesh B. Chowdhury, Ozgur Sinanoglu, Ramesh Karri, and Satwik Patnaik. ASCENT: Amplifying power side-channel resilience via learning and monte-carlo tree search. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2024.
5. Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, Zakaria Najm, et al. Pre-silicon power side-channel evaluation: Methodologies and automated tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3190–3203, 2020.
6. Roderick Bloem, Hannes Gross, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal verification of masked hardware implementations in the presence of glitches. In *EUROCRYPT 2018*, volume 10821 of *Lecture Notes in Computer Science*, pages 321–353. Springer, 2018.
7. Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of checking cryptographic protocols for faults. *Journal of Cryptology*, 14(2):101–119, 2001.
8. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29. Springer, 2004.
9. Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. In *Cryptology ePrint Archive, Report 2019/953*, 2019.
10. Yuchen Gao, Haocheng Ma, Junfeng Kong, Yiqiang Zhao, and Weikang Qian. EMSIM+: Accelerating electromagnetic security evaluation with generative adversarial network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2023.
11. Johannes Haring, Vedad Hadžić, and Roderick Bloem. Closing the gap: Leakage contracts for processors with transitions and glitches. In *IACR Transactions on Cryptographic Hardware and Embedded Systems (CHES)*, volume 2024, pages 110–132, 2024.
12. Meng He, Jinwoo Park, Adib Nahiyan, Apostol Vassilev, Yier Jin, and Mark Tehranipoor. RTL-PSC: Automated power side-channel leakage assessment at register-transfer level. In *IEEE VLSI Test Symposium (VTS)*, pages 1–6, 2019.
13. Pantea Kiaei and Patrick Schaumont. SoC root canal! root cause analysis of power side-channel leakage in system-on-chip designs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):751–773, 2022.
14. Pantea Kiaei, Yoo-Seung Won, Zhenyuan Liu, and Patrick Schaumont. Gate-level side-channel leakage ranking with architecture correlation analysis. *IEEE Transactions on Emerging Topics in Computing*, 12(2):496–507, 2024.

15. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Berlin, Heidelberg, 1999. Springer.
16. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
17. Asha Lakshmy, Chester Rebeiro, and Swarup Bhunia. FORTIFY: Analytical pre-silicon side-channel characterization of digital designs. In *Asia and South Pacific Design Automation Conference*, pages 660–665, 2022.
18. Zhenyuan Liu and Patrick Schaumont. Root-cause analysis of power-based side-channel leakage in lightweight cryptography candidates. In *Proceedings of the NIST 5th Lightweight Cryptography Workshop*, 2022.
19. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer Science & Business Media, 2007.
20. Walid El Maouaki, Alberto Marchisio, and Muhammad Shafique. Q-leak: Quantum-based leakage verification for side-channel countermeasures, 2026.
21. Patanjali Slpsk, Prasanna Karthik Vairam, Chester Rebeiro, and V. Kamakoti. KARNA: A gate-sizing based security aware EDA flow for improved power side-channel attack protection. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.
22. James E Stine et al. FreePDK: An Open-Source Variation-Aware Design Kit. In *IEEE International Conference on Microelectronic Systems Education*, pages 173–176, 2007.
23. Justin Thaler. *Proofs, Arguments, and Zero-Knowledge*. Foundations and Trends in Privacy and Security, 2022.
24. Mohit Tiwari, Hassan MK Wassel, Bita Mazloom, Roberto Guanciale, Frederic T Chong, and Timothy Sherwood. Complete information flow tracking from the gates up. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 109–120. ACM, 2009.
25. Riad S Wahby et al. Research directions for verifiable crypto-physically secure tees. *arXiv preprint arXiv:2410.03183*, 2024.
26. Yingchen Wang, Riccardo Paccagnella, Elizabeth T. He, Hovav Shacham, Christopher W. Fletcher, and David Kohlbrenner. Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86. In *USENIX Security*, pages 679–697, 2022.
27. Yue Yao, Tarun Kathuria, Begul Ege, Ognjen Glamocanin, Mirjana Stojilovic, and Patrick Schaumont. Architecture correlation analysis (ACA): Identifying the source of side-channel leakage at gate-level. In *IEEE Hardware Oriented Security and Trust (HOST)*, pages 188–196, 2020.
28. Loay Younes, Bilal Mohammad, Mahmoud Al-Qutayri, Hani Saleh, and Dima Kilani. Real-time switched capacitor based power side-channel attack detection. In *International Conference on Microelectronics (ICM)*, pages 253–257, Abu Dhabi, United Arab Emirates, 2023.
29. Ruiyi Zhang, Albert Cheu, Adria Gascon, Daniel Moghimi, Phillipp Schoppmann, Michael Schwarz, and Octavian Suciu. SNPeek: Side-channel analysis for privacy applications on confidential VMs. *arXiv preprint arXiv:2506.15924*, 2025.