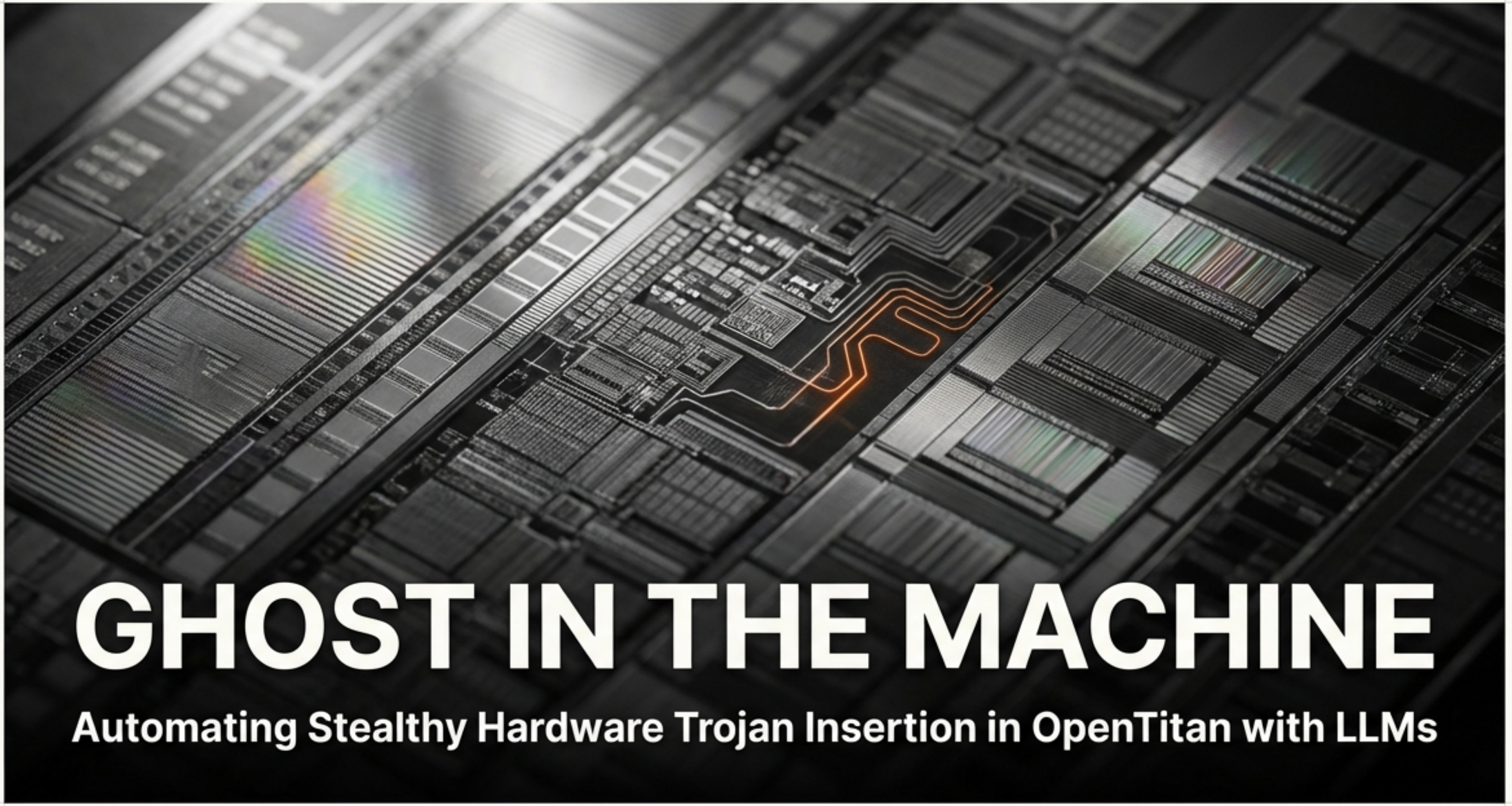
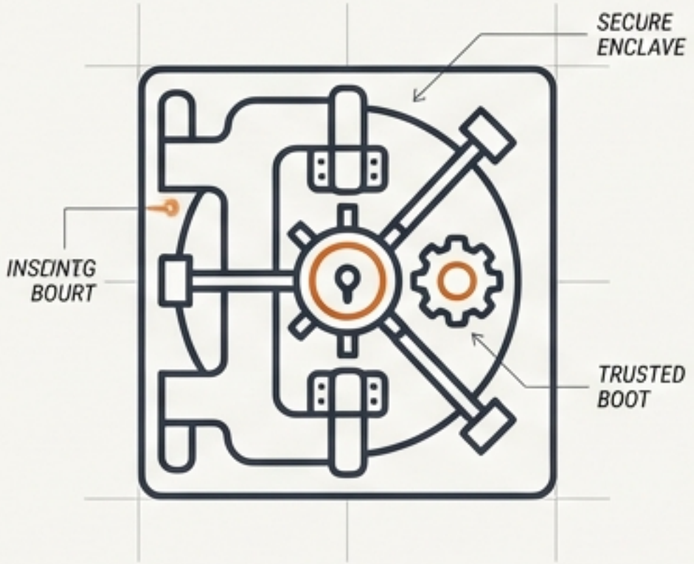
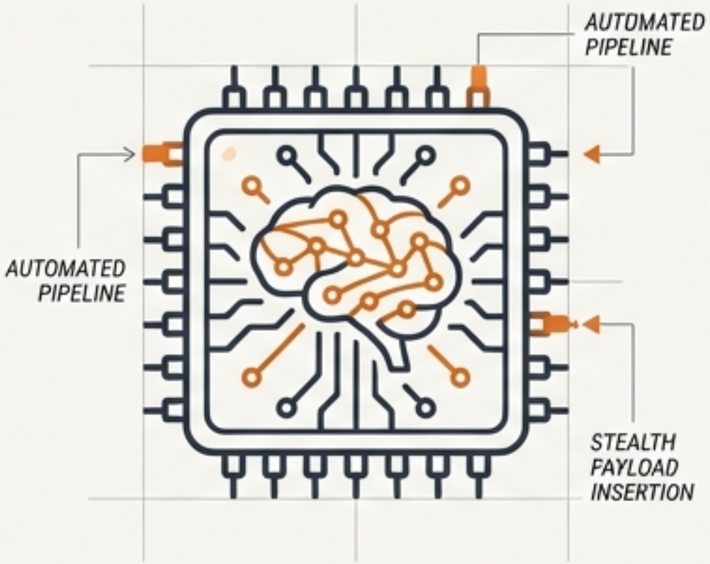

GHOST IN THE MACHINE
Automating Stealthy Hardware Trojan Insertion in OpenTitan with LLMs

A New Class of Hardware Threat Has Emerged



The Challenge

OpenTitan is a production-grade, open-source Root of Trust, making it a fortress of modern hardware security. Its large, highly verified codebase, strict coding guidelines, and extensive verification infrastructure make manual Trojan insertion exceptionally difficult and time-consuming.



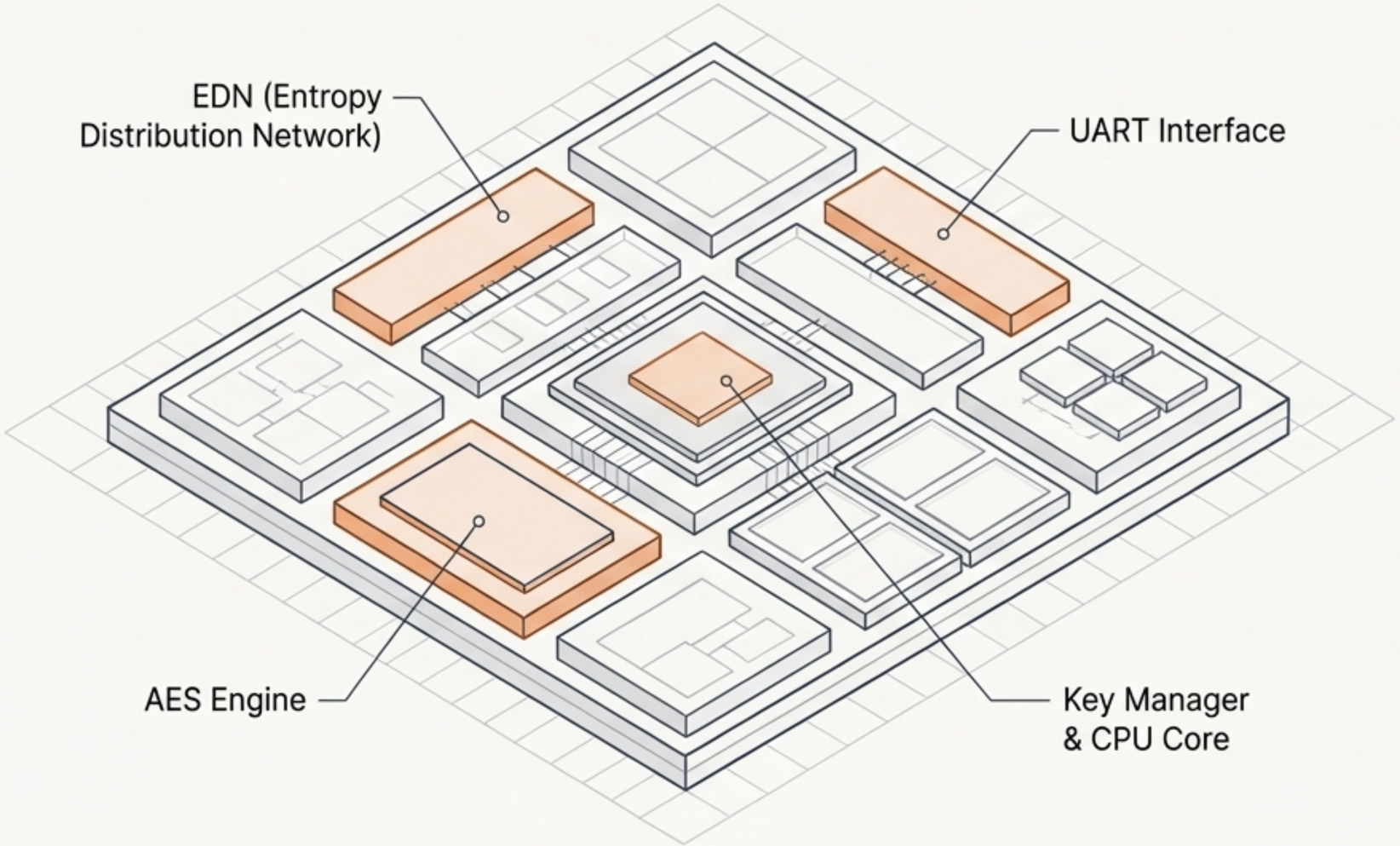
The Breakthrough

We used the GHOST LLM framework to create a fully automated pipeline for hardware Trojan insertion. This system successfully designed and integrated five unique, stealthy Trojans across OpenTitan's most critical modules.

The Implication

This demonstrates a new paradigm for hardware security threats. It also provides a powerful new capability for automated security validation and red-teaming to discover vulnerabilities before adversaries do.

The Target: Breaching a Production-Grade Root of Trust



Open-Source Root of Trust

The foundation for security in data centers and critical infrastructure.

Security-Hardened

Built with strict coding guidelines, built-in assertions, and extensive Design Verification (DV) infrastructure to prevent flaws.

Large & Hierarchical

Unlike small, isolated IP cores, OpenTitan is a complex System-on-Chip (SoC) with deep module interactions.

The GHOST Framework: An Automated Pipeline for Trojan Insertion

We used the GHOST framework, an LLM-driven system, to automate the entire Trojan development lifecycle. The process transforms high-level intent into synthesizable, integrated hardware modifications.

Identify

Analyze suitable modules (e.g., datapath, control-oriented components) for feasible insertion points based on observability and architectural impact.

Generate

Use prompt-driven generation to create synthesizable RTL for the Trojan logic, specifying trigger, payload, and style constraints.

Integrate

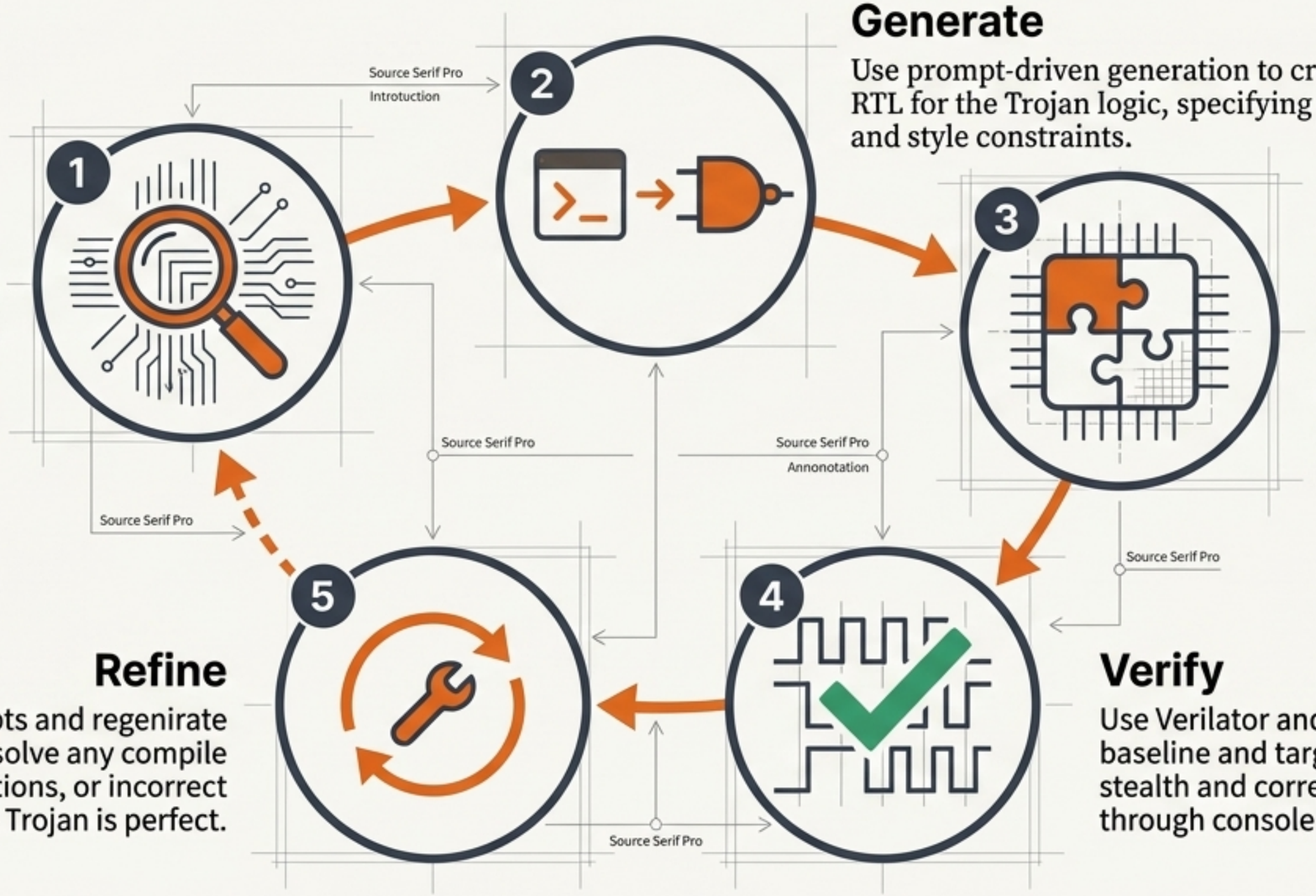
Insert the auto-generated logic into the target module's RTL, ensuring clocking, reset, and state machines remain consistent.

Verify

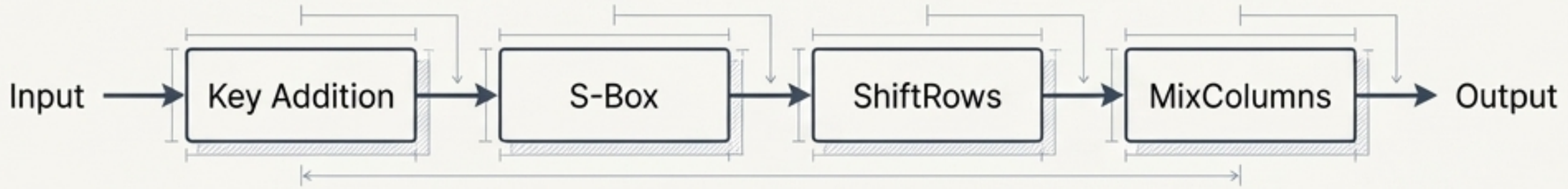
Use Verilator and Fusesoc to run baseline and targeted tests, confirming stealth and correct activation behavior through console logs.

Refine

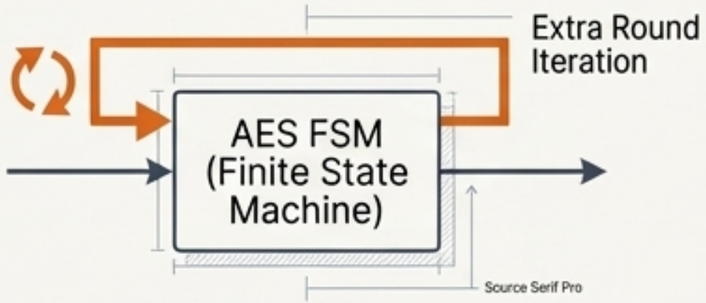
Iteratively adjust prompts and regenirate RTL patches to resolve any compile failures, style violations, or incorrect behaviors until the Trojan is perfect.



Corrupting the Crown Jewels: Targeting AES Cryptography



Round Repetition



Target

The AES core's main control Finite State Machine (FSM).

Trigger

A rare internal condition on the masked key input (`key_init_cipher[0][3:0] == 4'hA`).

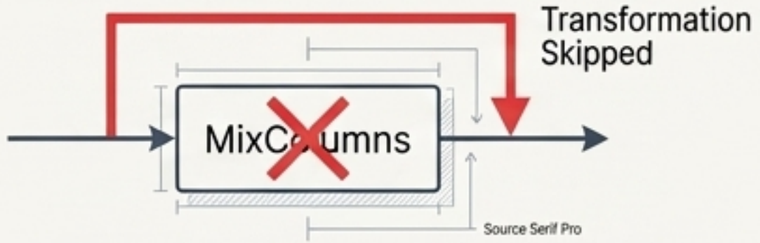
Vector

Introduces a single-cycle repetition of the `CRYPT` state, causing the datapath to execute one extra round iteration.

Payload

The FSM is forced to repeat the `CRYPT` state for one cycle, delaying output and subtly altering the ciphertext transformation.

Selective Fault Injection



Target

The `aes_mix_columns` module within the AES round function.

Trigger

An internal 20-bit counter reaches a rare value (`20'h8AD1E`), occurring approximately once every million cycles.

Vector

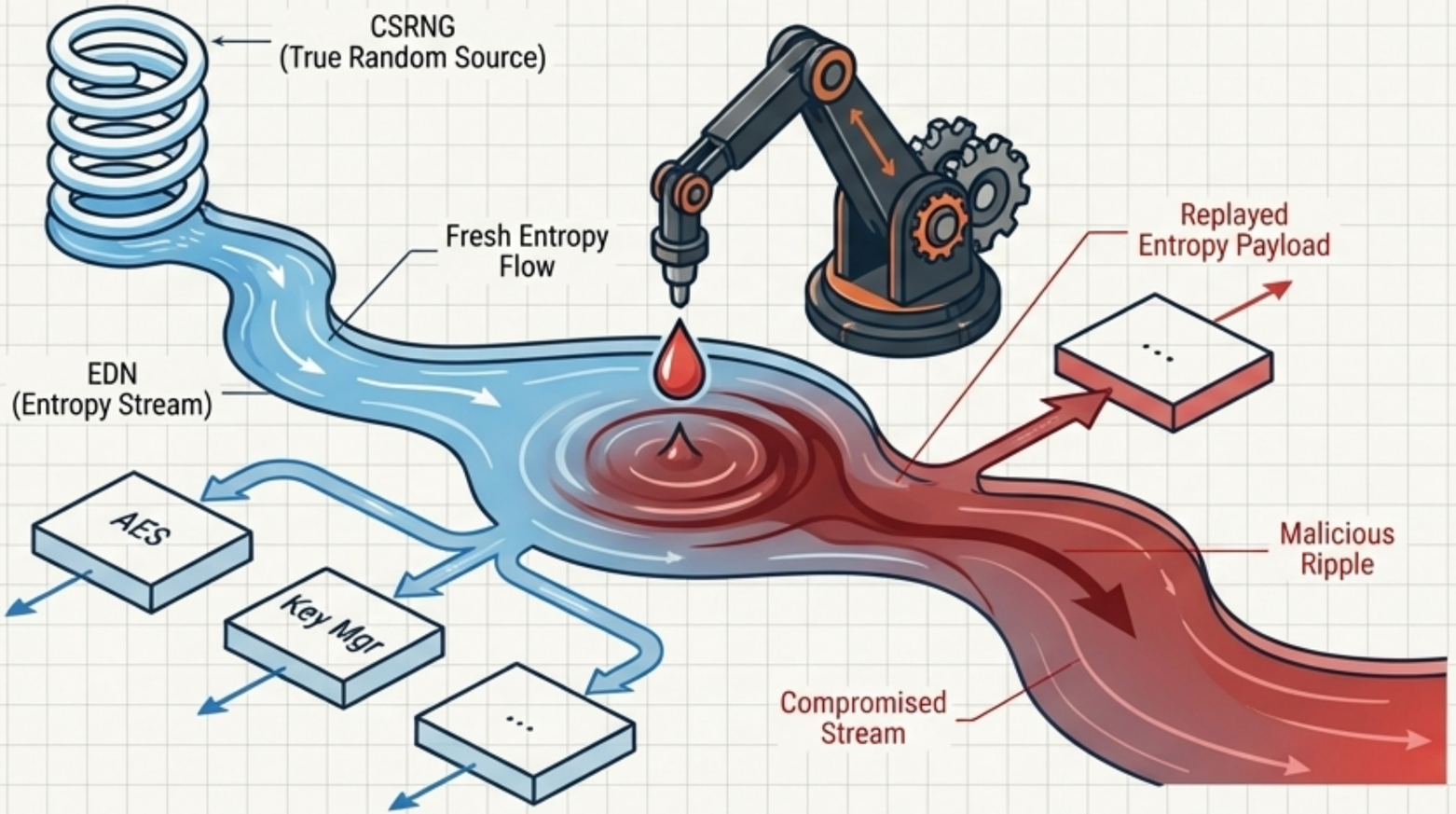
Performs a single-cycle bypass of the MixColumns transformation, a critical linear step in AES.

Payload

The module's output (`data_o`) is assigned its input (`data_i`), effectively skipping MixColumns for one cycle and creating a difficult-to-detect fault.

Poisoning the Well: A Critical Attack on the Entropy Source

The Entropy Distribution Network (EDN) supplies fresh, unique random numbers from the CSRNG to all cryptographic components. Replaying an entropy value, even once, can catastrophically weaken the entire system's security.



Trojan 3: Entropy Reuse

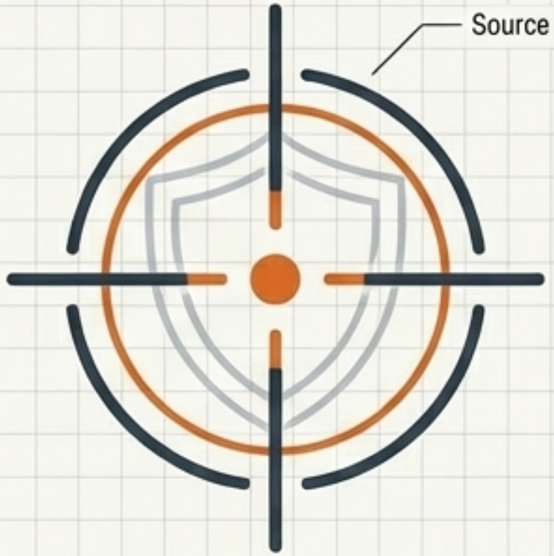
Target: The EDN core's entropy distribution path, specifically the CSRNG packer output.

Vector: The Trojan captures a previous entropy word and replays it exactly once when the trigger condition is met.

Trigger: A 16-bit internal counter that increments on each valid entropy consumption reaches `32767`, a value nearly impossible to hit during standard verification.

Payload: A mux selects a stored, old entropy value (`trojan_entropy_buffer_q`) instead of the real, fresh one for a single cycle, violating the core security promise of the EDN.

Takeaway: AI-Generated Trojans Can Weaken Core Security Functions with Surgical Precision



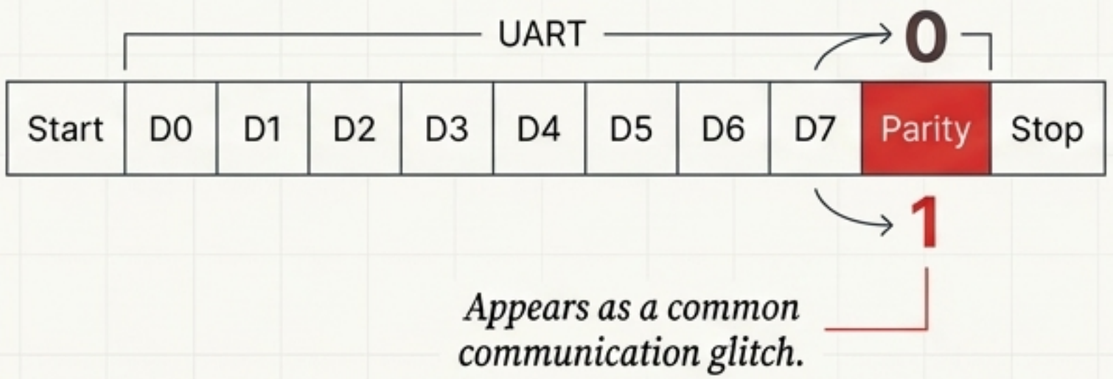
LLM-generated Trojans successfully targeted and weakened fundamental security modules like AES and EDN.

The attacks relied on ***minimal, single-cycle*** *manipulations (FSM stalls, datapath bypasses, value replays)* that are ***difficult to detect.***

These modifications were designed to be stealthy, *preserving all architectural interfaces* and *passing standard regression tests* without raising alerts.

Undermining the Perimeter: Attacking Communication Interfaces

Trojan 4: Parity Manipulation (Integrity Attack)



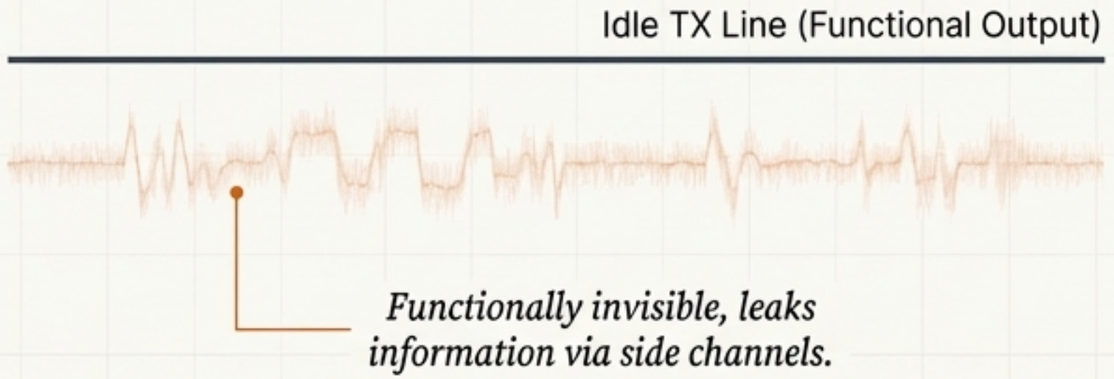
Target: The UART transmit datapath's parity generation logic.

Vector: Flips the parity bit for a single outgoing data frame. This attack appears as a common, innocent communication glitch.

Trigger: A secret three-byte sequence received on the UART RX line (`0xA7` → `0x3C` → `0xF1`).

Payload: The final calculated `tx_parity` bit is inverted for exactly one transmitted frame before the Trojan disarms itself.

Trojan 5: Idle Line Noise Injection (Side-Channel Attack)



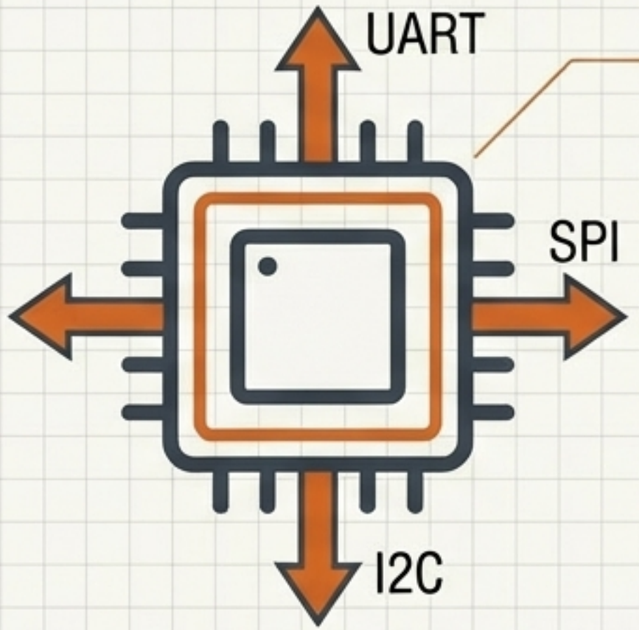
Target: The UART transmit path during idle periods.

Vector: Generates non-functional, pseudo-random switching activity internally. The payload is functionally invisible but creates a measurable power/EM side-channel signature.

Trigger: A rare combination of the TX path being idle and a specific pattern appearing in the RX history buffer (`rx_val_q[15:12] == 0xA` and `rx_val_q[3:0] == 0x5`).

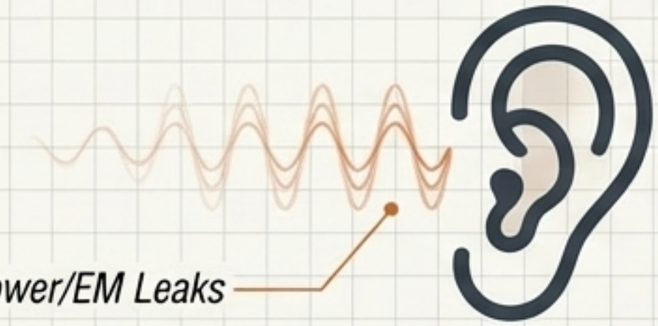
Payload: A 4-bit shift register (`trojan_sr_q`) is activated, creating internal signal toggling that is not routed to any functional output.

Takeaway: Automated Trojans Can Create Ambiguous Faults and Hidden Side Channels



The ***GHOST framework*** can craft Trojans that target a system's external interfaces (like UART) with high precision.

These attacks can manifest as faults that ***mimic natural communication errors*** (parity flips), making them extremely difficult to attribute to malicious intent.



Furthermore, Trojans can be designed to be ***functionally harmless*** while generating hidden information leaks through side channels (power/EM), rendering them invisible to standard verification.

The Universal Signature of Success: Perfect Stealth

Every Trojan was designed to be invisible to OpenTitan's extensive regression test suite. They only revealed themselves when a specific, targeted testbench was used to activate their hidden trigger.

Trojan	Target Module	Standard Regression Tests	Targeted Trojan Test
Round Repetition	`aes_core`	✓ PASSED	✗ DETECTED
Selective Fault Injection	`aes_mix_columns`	✓ PASSED	✗ DETECTED
Entropy Reuse	`edn_core`	✓ PASSED	✗ DETECTED
Parity Manipulation	`uart_core`	✓ PASSED	✗ DETECTED
Idle Line Noise Injection	`uart_core`	✓ PASSED	✗ DETECTED

The LLM as a Malicious Co-Designer

The GHOST framework did more than just write code; it acted as a partner that understood the complex constraints of a high-assurance design environment. Its output demonstrated a nuanced understanding of hardware design principles.

Produced Synthesizable RTL

Generated logic that was not just syntactically correct, but structurally sound for hardware synthesis.

Preserved OpenTitan's Coding Style

Adhered to strict conventions like `always_ff`/`always_comb` separation, synchronous resets, and masking discipline.

Minimized Manual Intervention

The generated RTL 'integrated cleanly on the first attempt, with only minor adjustments needed' and 'reduced manual editing.'

Maintained System Integrity

Crafted Trojans that avoided disrupting state machines, datapaths, and critical timing paths.

A Dual-Use Technology: The Future of Hardware Attack and Defense

The Threat: An Adversarial Co-Pilot

Automated, AI-driven Trojan insertion represents a significant **new threat vector.** It lowers the barrier to entry for sophisticated hardware attacks and can create **vulnerabilities** that are nearly impossible to find with traditional methods.

The Opportunity: An Automated Red Team

This same technology is a powerful tool for defenders. It enables automated security validation at a scale and speed previously unimaginable, allowing designers to find and fix vulnerabilities by thinking like an AI-powered attacker.

Key Questions for the Community

- How must our verification strategies evolve to defend against AI-generated hardware threats?
- How can we responsibly leverage these techniques to build more resilient and secure systems from the ground up?

The Ghost is Real: AI Can Autonomously Infiltrate Secure Hardware

1

Automation is Viable

LLM-based frameworks can successfully automate the insertion of stealthy Trojans into complex, production-grade SoCs like OpenTitan.

2

Stealth is Achievable

These Trojans were designed to be invisible to standard verification, passing rigorous regression tests while remaining precisely triggerable.

3

The Landscape has Changed

This capability fundamentally alters the hardware security landscape, presenting both a formidable new class of threats and an unprecedented opportunity for automated defense.