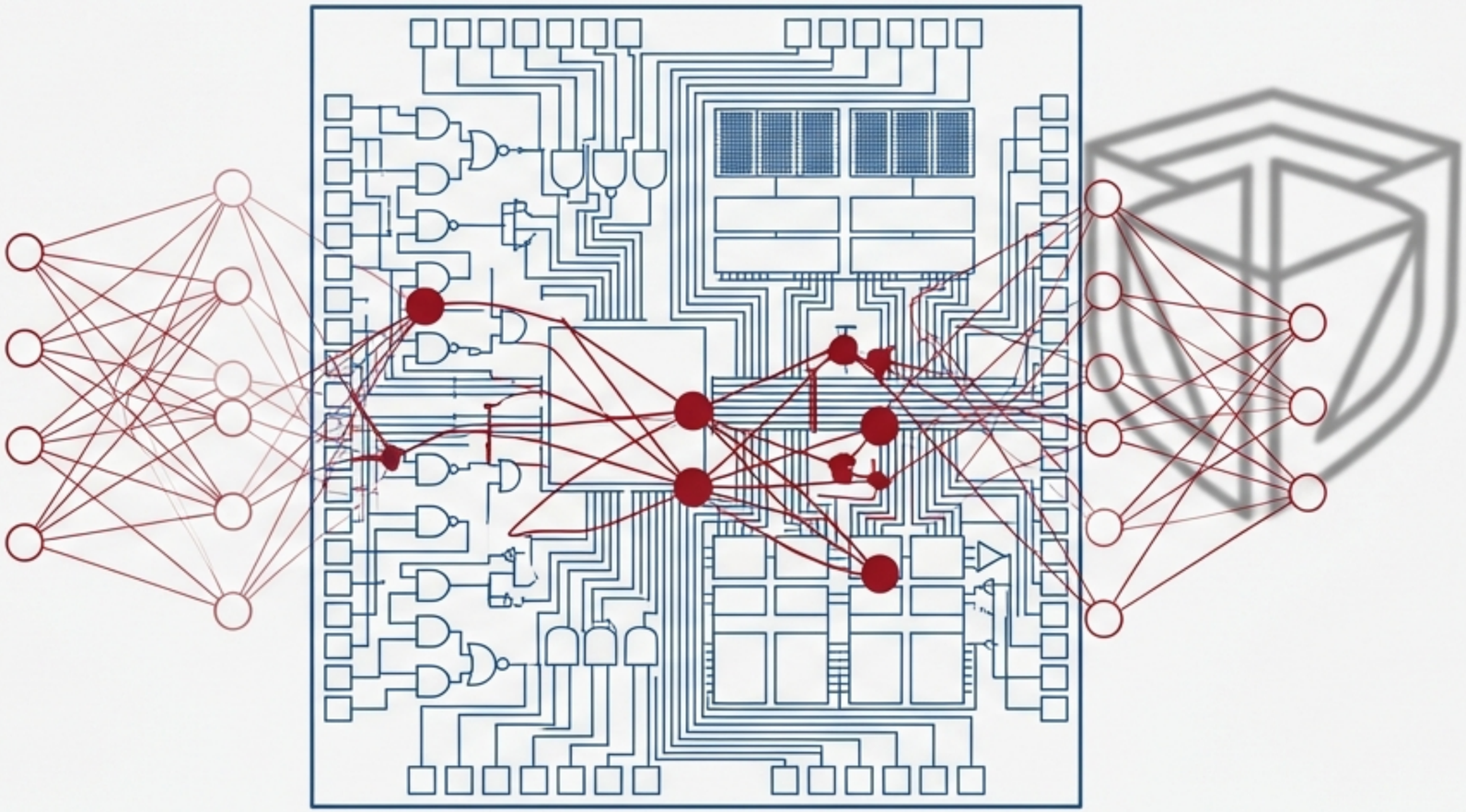
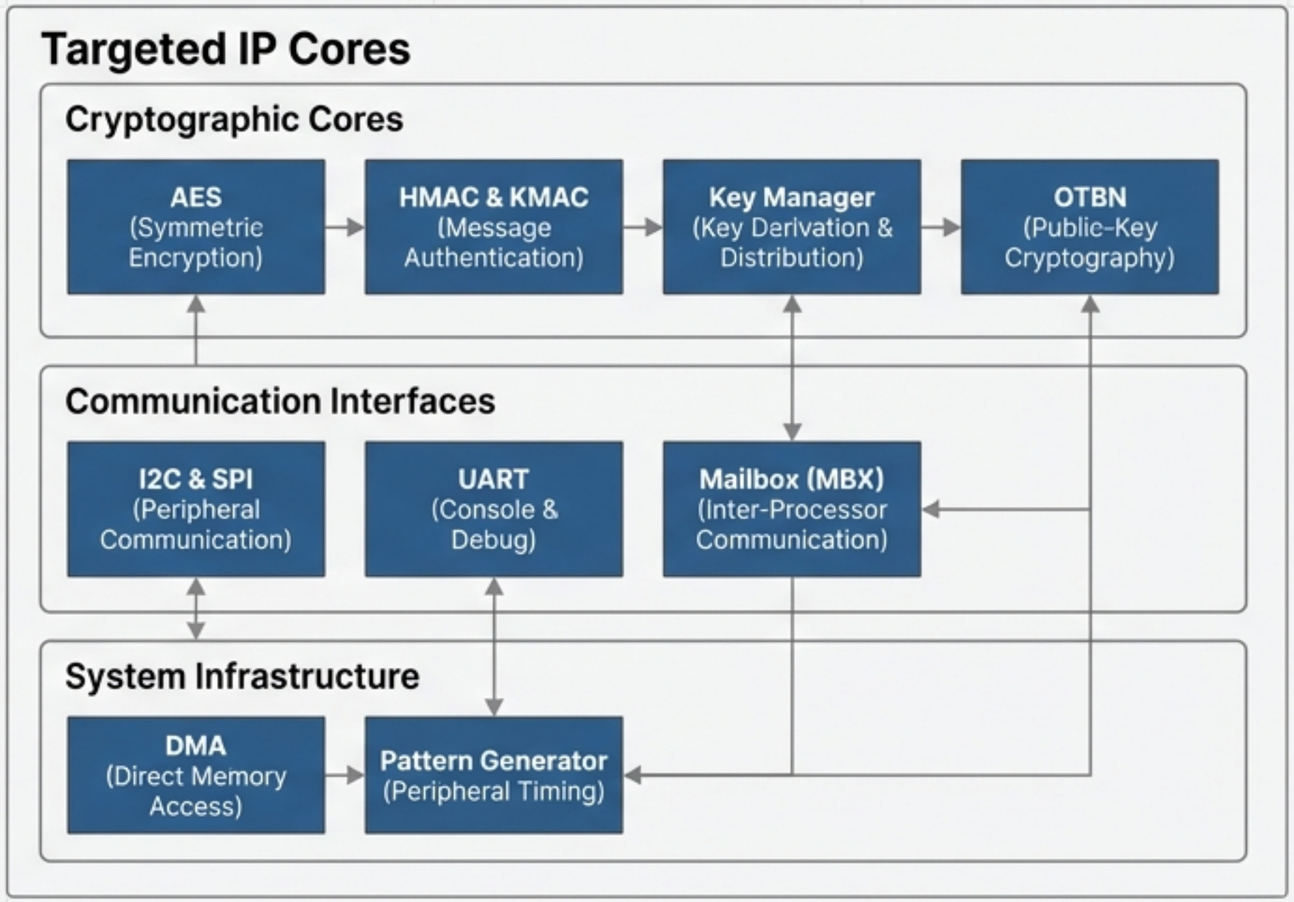
Weaponizing AI: Automated Hardware Trojan Insertion in OpenTitan

A report on systematically compromising secure silicon using Large Language Models.



The Target: A State-of-the-Art Secure Silicon Platform

OpenTitan is an open-source project for building transparent, high-quality, and secure silicon. Its design is a benchmark for trustworthy hardware. Our objective was to probe its defenses by targeting a wide array of its most critical components.



The Automated Arsenal: An LLM-Powered Attack Pipeline

We developed a semi-automated workflow that leverages the GHOST framework for RTL manipulation and a Large Language Model (GPT-4.1) for strategic guidance. This pipeline accelerates the entire process from vulnerability discovery to stealthy Trojan generation.

Module Selection & Analysis

LLM interprets RTL to understand control signals, datapaths, and security boundaries across the 11 target modules. Proposes architectural insertion points that minimize detectability.

Automated Trojan Generation (GHOST + LLM)

LLM generates concepts for triggers and payloads (counters, FSMs, covert channels). GHOST synthesizes the Verilog for insertion, creating logic for pattern detection, state machines, and output overrides.

Stealth Validation

OpenTitan's standard smoke tests are used to verify that the chip's baseline functionality remains unchanged, confirming pre-trigger stealth. The LLM helps prove the absence of functional interference before activation.

An Attacker's Playbook: Three Vectors of Compromise

The automated pipeline successfully generated a diverse catalogue of Trojans. These can be categorized into three fundamental attack strategies, each demonstrated with specific examples in the following sections.

Part 1: Silent Corruption

Trojans that subtly alter functionality, modify data, or corrupt memory transactions. The system appears to work correctly, but its integrity is compromised.

Part 2: Covert Exfiltration

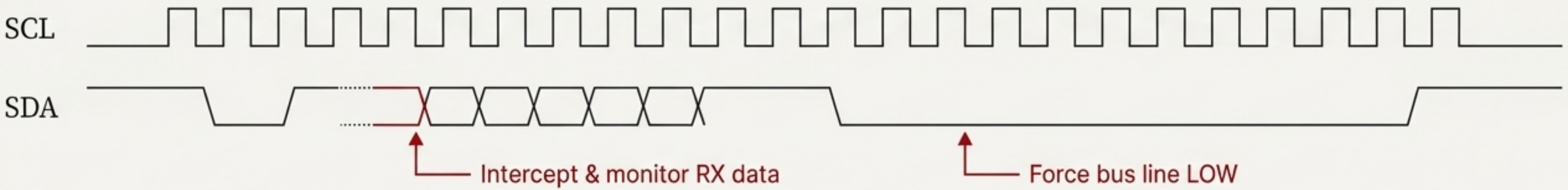
Trojans designed to leak sensitive information. They create hidden channels to exfiltrate critical data like cryptographic keys and internal system state.

Part 3: Total Annihilation

Trojans that cause a permanent Denial of Service. Once triggered, they irreversibly disable critical system components, rendering the hardware useless until a full reset.

Part 1: Silent Corruption

Manipulating Peripheral Communication: I2C



Trojan Profile 1

Target Module
`i2c_fifos.sv`

Trigger
A specific 5-byte sequence written consecutively to the RX FIFO: `{0x09, 0x04, 0x00, 0x02, 0x16}`.

Payload
A flag is permanently set, causing all subsequent data read from the RX FIFO to have its bits reversed (`bit[7:0] → bit[0:7]`). This persistently corrupts all received I2C data.

Stealth
Target test passed. The specific trigger sequence is not covered in standard verification.

Trojan Profile 2

Target Module
`i2c_core.sv`

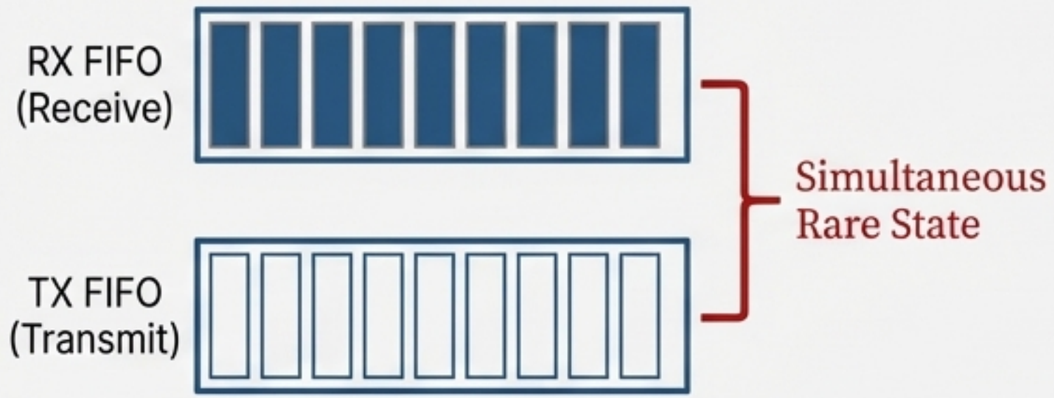
Trigger
A rare override state (`sclval=1, sdaval=1, host_enable=1`) is held for 8 consecutive clock cycles.

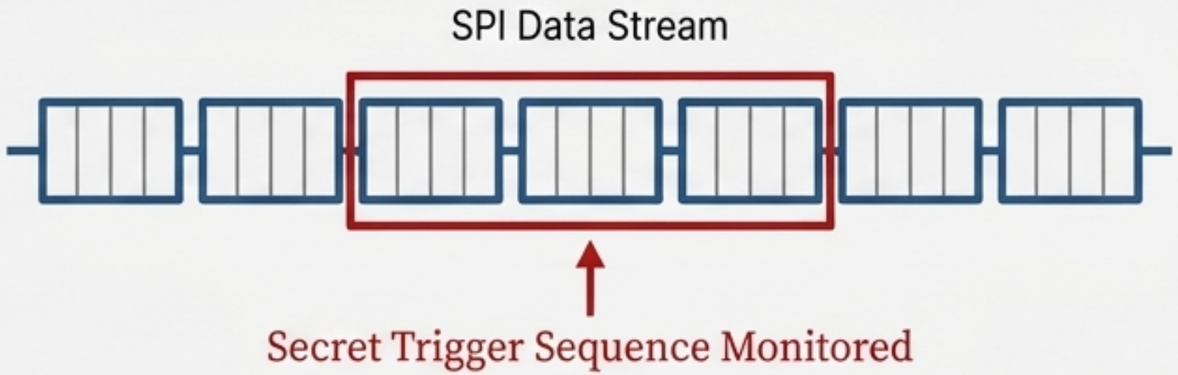
Payload
After 8 cycles, the `sda_o` output is forcibly driven LOW, creating a permanent bus fault and disrupting all I2C communication.

Stealth
Target tests passed. This specific override condition is not exercised in tests.

Part 1: Silent Corruption

Corrupting Serial Data Streams: UART & SPI





Trojan Profile 1: UART

Target Module: `uart_core.sv`

Trigger: A rare state where the TX FIFO is completely empty *while* the RX FIFO is simultaneously full.

Payload: For a window of 8 clock cycles, the Most Significant Bit (MSB) of all received UART data (`hw2reg.rdata.d[7]`) is inverted, creating subtle bit-flip errors.

Stealth: Smoke tests passed. The simultaneous empty/full FIFO condition is not created during standard tests.

Trojan Profile 2: SPI

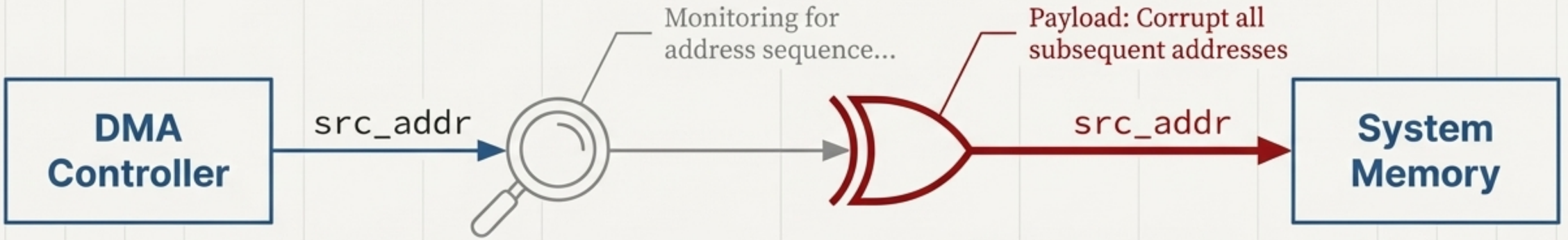
Target Module: `spi_host_data_fifos.sv`

Trigger: A specific 5-byte sequence received consecutively in the RX data stream: `{0x09, 0x04, 0x00, 0x02, 0x16}`.

Payload: A permanent flag is set, causing all subsequent `rx_data_o` outputs to have their bits reversed. This corrupts all received SPI data until reset.

Stealth: Smoke tests passed. This byte sequence is not included in tests.

Compromising System Memory: DMA



Trojan Profile

Target Module: `dma.sv`

Trigger: The DMA controller performs three consecutive read operations from a specific sequence of source addresses:

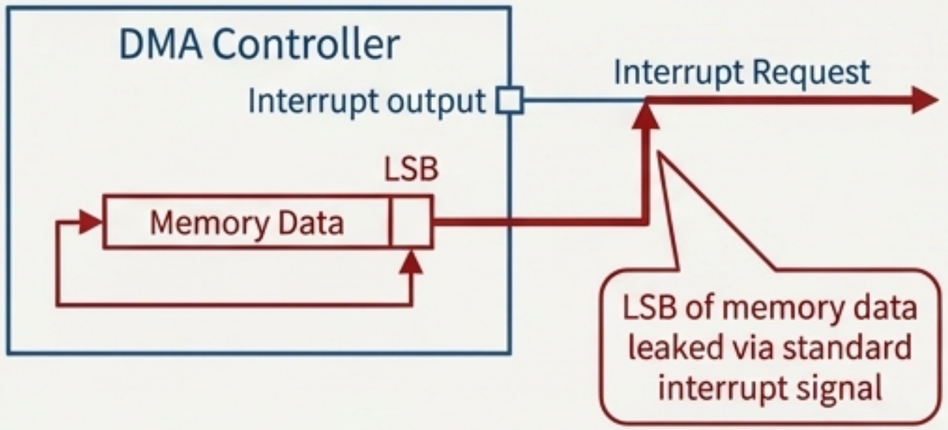
1. `0xCAFE0001`
2. `0xDEAD0002`
3. `0xBEEF0003`

Payload: A flag is permanently set. All subsequent bus addresses generated by the DMA are XORed with a fixed pattern. This corrupts all future memory transactions, leading to system-wide data corruption or a denial of service.

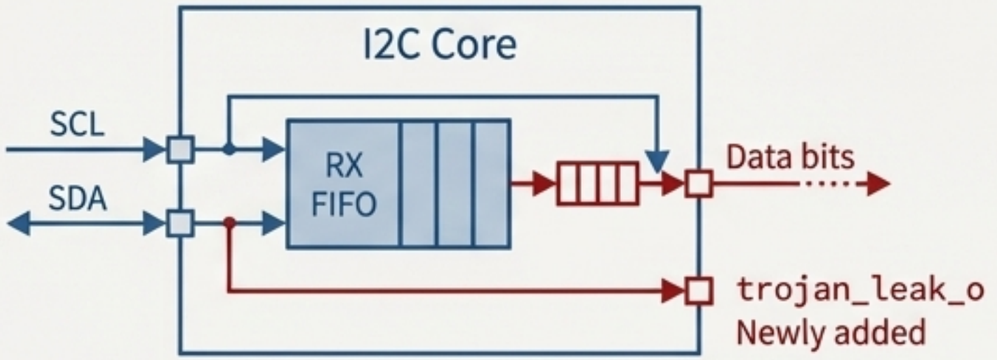
Part 2: Covert Exfiltration

Turning System Signals into Covert Channels: DMA & I2C

DMA Exfiltration



I2C Exfiltration



Trojan Profile 1: DMA

Target Module: `dma.sv`

Trigger: The lower 8 bits of the `lsio_trigger_i` input match the pattern `8'b10101010`.

Payload: A leak state is permanently set. The Least Significant Bit (LSB) of memory read data is exfiltrated, one bit at a time, via a standard interrupt output channel.

Trojan Profile 2: I2C

Target Module: `i2c_core.sv`

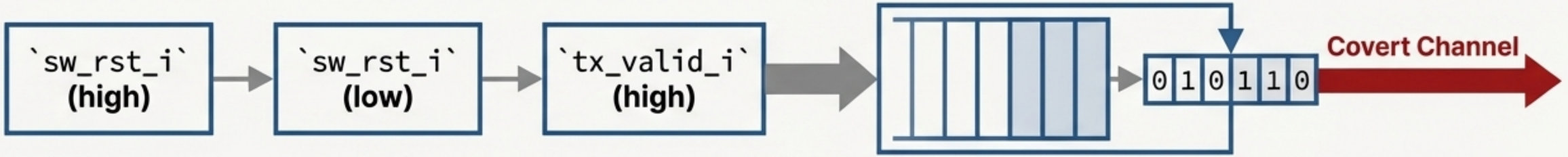
Trigger: A rare XOR combination of two internal sampling registers (`scl_rx_val ^ sda_rx_val`) matches the pattern `16'hA5C3`.

Payload: The 8 bytes of data from the RX FIFO are captured and leaked bit-by-bit over a dedicated covert output pin (`trojan_leak_o`).

Stealth: Target tests passed. The covert output pin is not monitored during verification.

Part 2: Covert Exfiltration

Leaking Internal State: SPI FIFO Depth



Trojan Profile

Target Module: `spi_host_data_fifos.sv`

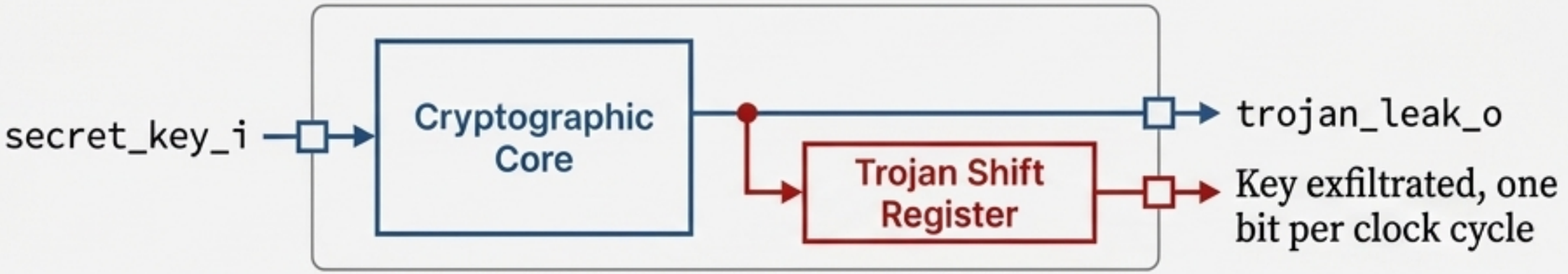
Trigger: A rare, three-phase event sequence:
1. Software reset (`sw_rst_i`) asserts.
2. Software reset deasserts.
3. Transmit valid (`tx_valid_i`) asserts immediately after.

Payload: The internal TX FIFO depth (`tx_qd_o`) is captured. The value is then leaked serially, bit-by-bit, over a dedicated covert output channel (`trojan_covert_out`). This reveals internal buffer occupation.

Stealth: Smoke tests passed. This specific reset-then-write sequence is not part of standard testing.

Part 2: Covert Exfiltration

The Ultimate Prize: Exfiltrating Cryptographic Keys from HMAC & KMAC



Trojan Profile 1: HMAC

Target Module: `hmac_core.sv`

Trigger: The `message_length_i` input matches the specific 32-bit pattern `0xACEDCAFE`.

Payload: The `secret_key_i` is captured and leaked serially over a covert output pin, one bit per clock cycle.

Trojan Profile 2: KMAC

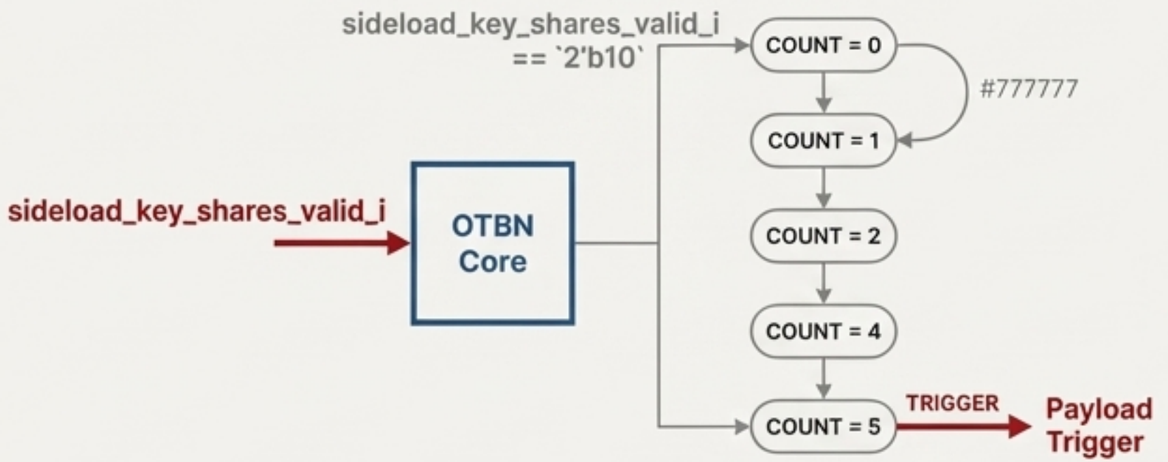
Target Module: `kmac_core.sv`

Trigger: A rare control sequence: the `process_i` signal is asserted for exactly 10 consecutive pulses, immediately followed by an assertion of the `start_i` signal.

Payload: The `secret_key_i` is captured and leaked serially over a covert output pin, exfiltrating the key one bit at a time.

Part 2: Covert Exfiltration

Compromising Asymmetric and Symmetric Crypto: OTBN & AES



Trojan Profile 1: OTBN

Target Module: `otbn_core.sv`

Trigger: The `sideload_key_shares_valid_i` signal is held at the specific binary pattern `2'b10` for exactly 5 consecutive clock cycles.

Payload: The first byte of the sideloaded key share (`sideload_key_shares_i[0][7:0]`) is captured and leaked serially over a covert output pin.

Trojan Profile 2: AES Core

Target Module: `aes_core.sv`

Trigger: The plaintext input (`data_in_flat`) matches a secret `TROJAN_PATTERN` *and* the encryption `start` signal is asserted simultaneously.

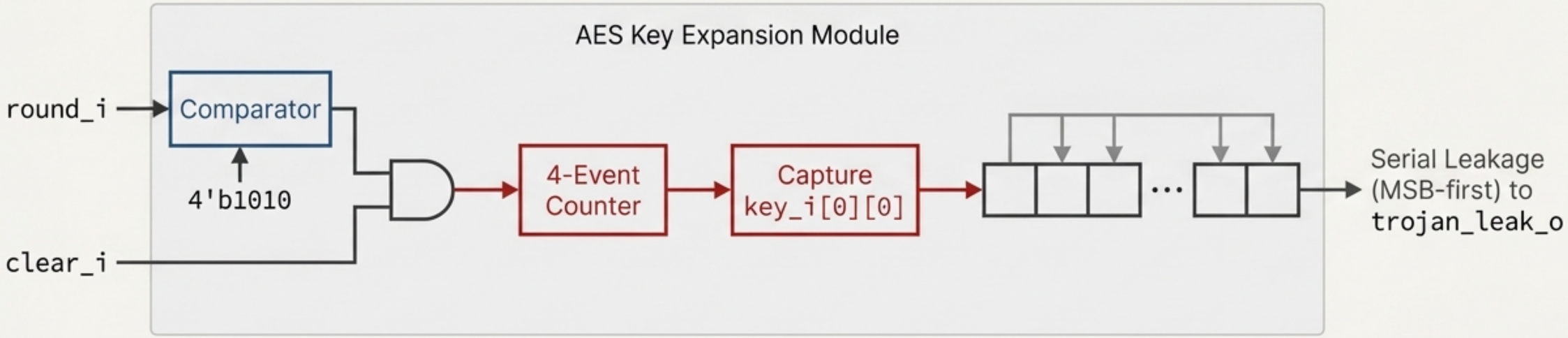
Payload: The AES key is captured. One bit of the key is leaked over a covert output pin (`trojan_covert_o`) each time an encryption cycle completes.

Stealth: Smoke tests passed. The specific `TROJAN_PATTERN` **is not used** in verification tests.

Editorial Intelligence Briefing

Part 2: Covert Exfiltration

A Surgical Strike: Leaking the AES Key Expansion



Trojan Profile

Target Module: `aes_key_expand.sv`

Trigger: An extremely rare sequential pattern: the `clear_i` signal pulses while the internal `round_i` signal is equal to `4'b1010`. This exact event must occur 4 consecutive times.

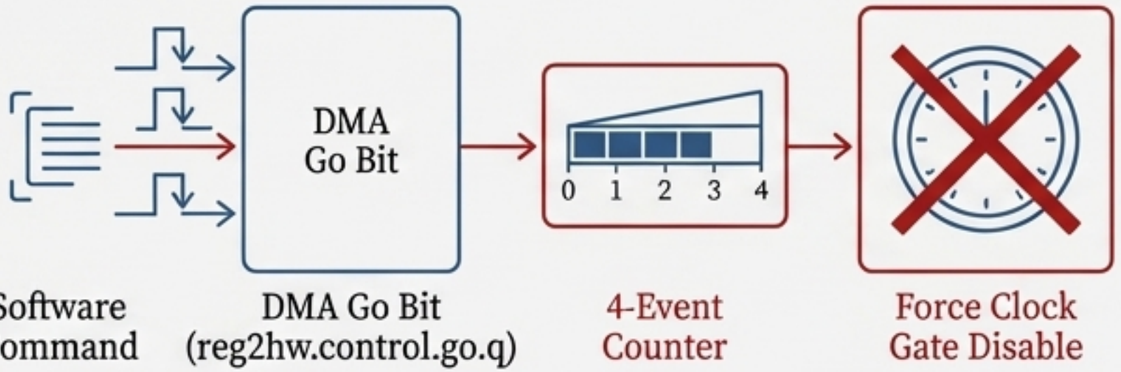
Payload: After the fourth occurrence, the first word of the AES key (`key_i[0][0]`) is captured. The 32 bits of the key word are then leaked serially, MSB-first, over a dedicated covert output pin (`trojan_leak_o`).

Stealth: Smoke tests passed. The trigger condition is highly unlikely to be met during standard verification.

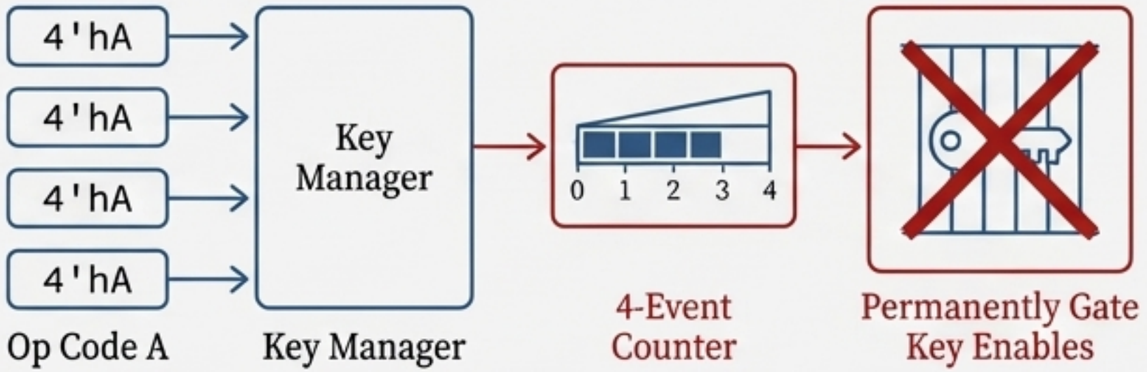
Part 3: Total Annihilation

Triggering Irreversible Shutdown via Software: DMA & Key Manager

DMA Shutdown Trigger



Key Manager Shutdown Trigger



Trojan Profile 1: DMA

- **Target Module:** `dma.sv`
- **Trigger:** Four rising edges are detected on the software-controllable DMA **go** bit (`reg2hw.control.go.q`).
- **Payload:** A permanent latch sets, forcing the core clock gating enable signal **LOW**. This halts all DMA logic and internal state updates, bricking the module until a hardware reset.

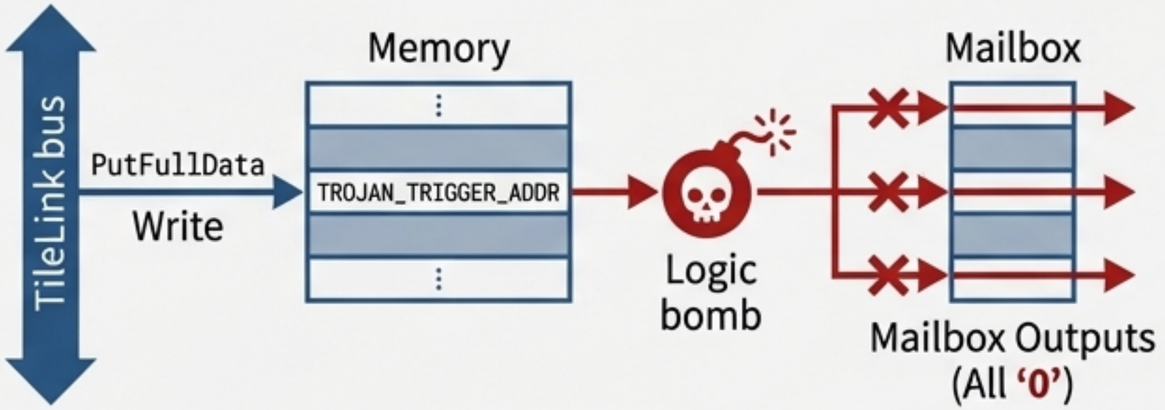
Trojan Profile 2: Key Manager

- **Target Module:** `keymgr.sv`
- **Trigger:** The software issues the same operation code (`4'hA`) four consecutive times.
- **Payload:** A trigger counter reaches its threshold, **permanently gating** all critical key manager enable signals. Key generation is **disabled**, and persistent error signals are forced active, **shutting down** the most sensitive component in the system.

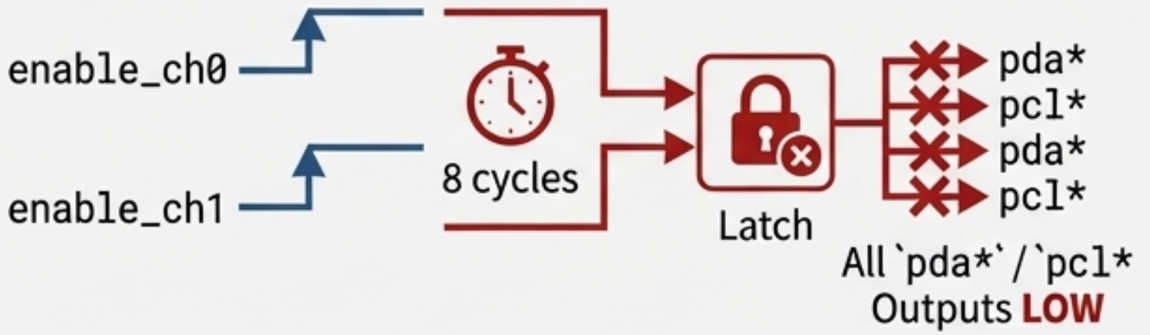
Part 3: Total Annihilation

Sabotaging Communication and Control: Mailbox & Pattern Generator

Mailbox Sabotage Trigger



Pattern Generator Sabotage Trigger



Trojan Profile 1: Mailbox (MBX)

Target Module: `mbx.sv`

Trigger: A TileLink write operation (`PutFullData`) is performed to a secret, undocumented trigger address (`TROJAN_TRIGGER_ADDR`).

Payload: A permanent flag sets, forcing all mailbox output buses to zero, disabling interrupts, and blocking internal SRAM transfers. Communication is severed until reset.

Trojan Profile 2: Pattern Generator

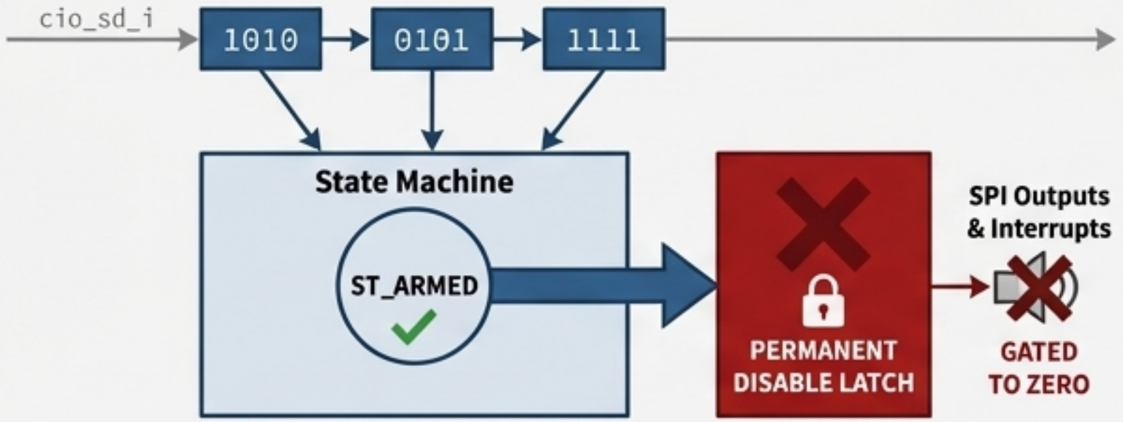
Target Module: `pattgen_core.sv`

Trigger: Both channel enable signals (`enable_ch0` and `enable_ch1`) are asserted simultaneously and held for 8 consecutive clock cycles.

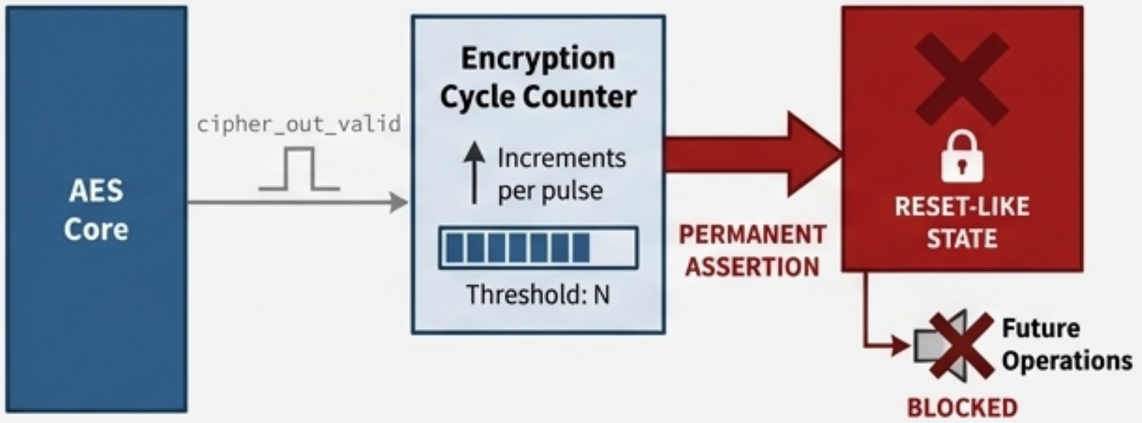
Payload: A permanent disable latch sets, forcing all pattern and clock outputs (`pda*`, `pcl*`) LOW and disabling all event signals. The module is rendered inert.

Part 3: Total Annihilation
Disabling Peripherals and Proving Effectiveness: SPI & AES

SPI Shutdown Trigger



AES Core Shutdown Trigger



Trojan Profile 1: SPI

Target Module: `spi_device.sv`

Trigger: A three-phase sequential pattern on the SPI data input (`cio_sd_i`) over three consecutive cycles: 4'b1010 → 4'b0101 → 4'b1111.

Payload: A state machine transitions to an armed state, setting a permanent disable flag. All SPI outputs, interrupts, and alerts are gated to zero.

Stealth: Smoke tests passed. This specific input data sequence is not used in DV.

Trojan Profile 2: AES Core

Target Module: `aes_core.sv`

Trigger: The completion of N valid encryption operations.

Payload: A counter reaches the threshold N, setting a permanent disable flag that forces the AES core into a reset-like state, blocking all future operations.

Validation: This is a key result. Base functionality passed smoke tests. When the trigger threshold was lowered for testing, the DoS behavior was **successfully observed and confirmed in simulation.**

The New Frontier: AI-Accelerated Hardware Threats

This work demonstrates that Large Language Models can automate the creation of sophisticated, stealthy hardware Trojans across a wide range of critical IP cores.

Accelerated Attack Prototyping

- LLMs dramatically reduce the manual effort required to understand complex RTL and devise trigger conditions.
- What once required weeks of expert analysis can now be prototyped in a fraction of the time.

Beyond Conventional Verification

- The generated Trojans rely on rare, sequential, and data-dependent triggers that are intentionally designed to fall outside the scope of standard functional tests like smoke tests.
- This highlights a fundamental gap in conventional pre-silicon verification methodologies.

A Paradigm Shift for Hardware Security

- The ability to programmatically generate stealthy hardware attacks requires a corresponding evolution in defensive strategies.
- Future security efforts must account for an adversary armed with AI-driven tools, demanding new approaches to verification, formal methods, and run-time monitoring.