# Automating LLMs for Hardware Trojan Insertion in OpenTitan

**Buddhi Perera**
**Milan Patel**

---

## 1. Introduction / Overview

This report presents the design, automated insertion, and analysis of several hardware Trojans introduced into OpenTitan IP cores using a combination of programmatic tools and large language models (LLMs). The project explores how Trojans can be engineered to remain dormant during normal operation yet produce destructive or malicious effects once triggered.

The Trojans developed in this study span multiple classes:

- **Denial of Service (DoS)**
- **Sensitive information leakage**
- **Functionality modification**
- **Performance degradation**

These Trojans were embedded into OpenTitan's **AES, I2C, UART, SPI, DMA, HMAC, KMAC, OTBN, Key Manager, Pattern Generator, and Mailbox (MBX)** modules—eleven subsystems representative of modern SoC security boundaries and critical operational infrastructure.

### Target Modules Overview

**Cryptographic Cores:**

- **AES** — Protects symmetric encryption operations; compromise enables plaintext recovery or key extraction
- **HMAC** — Provides message authentication; vulnerabilities allow signature forgery
- **KMAC** — Implements SHA-3 based keyed hashing; critical for authenticated encryption schemes
- **Key Manager** — Centralized key derivation and distribution; the most sensitive component in the system
- **OTBN** — Big Number accelerator for public-key cryptography; handles RSA/ECC operations

**Communication Interfaces:**

- **I2C** — Handles secure peripheral communication with sensors and TPMs
- **UART** — Serves debugging, boot integrity, and console communication roles
- **SPI** — High-speed peripheral interface for flash memory and secure elements
- **Mailbox (MBX)** — Inter-processor communication channel for secure message passing

**System Infrastructure:**

- **DMA** — Direct memory access controller; compromise enables unauthorized memory access
- **Pattern Generator** — Generates timing patterns for external peripherals; critical for protocol compliance

This selection makes them ideal candidates for exploring Trojan insertion in realistic secure hardware, covering the full spectrum from cryptographic primitives to system-level communication and memory management.

---

# Objectives

The primary objectives of this work were to:

1. **Identify suitable IP modules** (AES, I2C, UART, SPI, DMA, HMAC, KMAC, OTBN, Key Manager, Pattern Generator, MBX) for Trojan insertion based on criticality, internal signal visibility, and vulnerability potential.
2. **Use automated LLM-powered tools**, such as GHOST, to generate hardware Trojans and integrate them into RTL with minimal manual effort.
3. **Validate that pre-trigger functionality is preserved**, primarily through OpenTitan smoke tests and functional verification.
4. **Analyze Trojan triggers, payload implementations, stealth characteristics, and activation behavior** across different module types and attack vectors.
5. **Evaluate the role of GPT-4.1 and Claude Sonnet 4.5** in understanding RTL, proposing insertion points, generating trigger logic, and validating safety and stealth properties.

Through these activities, this report demonstrates how LLMs can accelerate hardware security research, enabling rapid prototyping of sophisticated Trojans that would otherwise require extensive manual RTL engineering across diverse IP categories.

---

# 2. Automated System & Workflow Overview

This section combines the toolset and workflow description into a unified explanation of the automation pipeline used to generate and evaluate Trojans.

## 2.1 Tools Used

**GHOST Trojan-Insertion Framework**

GHOST is a semi-automated tool for synthesizable hardware Trojan insertion. It was used to:

- Identify structurally suitable insertion points in OpenTitan RTL
- Generate custom trigger and payload logic
- Insert counters, accumulators, state machines, and override circuits
- Produce synthesizable Verilog modifications without altering primary datapaths

**Notable supported Trojan types utilized in this project:**

- **DoS** (AES, SPI, DMA, Key Manager, Pattern Generator, MBX)
- **Information leakage** (I2C, SPI, HMAC, KMAC, OTBN, AES, DMA)
- **Functionality modification** (DMA, I2C, UART, SPI)

**LLM Models Used (GPT-4.1)**

LLMs were extensively used to:

- Interpret complex RTL control signals and datapath interactions across cryptographic, communication, and system-level modules
- Propose architectural insertion points that minimize detectability in security-critical components
- Generate Trojan code: counters, shift registers, FSMs, trigger conditions, and covert channels
- Explain protocol-level behavior:
    - I2C START/STOP conditions and bus arbitration
    - UART FIFO watermarks and flow control
    - SPI data phase timing and chip select sequences
    - AES/HMAC/KMAC FSM transitions and round completion
    - DMA transfer state machines and address generation
    - Key Manager operation sequences and key derivation stages
    - OTBN instruction execution and sideload key handling
- Validate that inserted logic remains timing- and synthesis-safe
- Prove the absence of functional interference pre-trigger across all modules

The LLMs were critical for bridging the gap between design intent and stealth engineering, particularly for understanding the security implications of trigger placement in cryptographic cores.

**OpenTitan DV Smoke Tests**

OpenTitan's smoke tests—run using Verilator—were used to:

- Validate that base functionality remained correct across all eleven modified modules
- Detect regressions caused by faulty Trojan insertions
- Confirm that Trojan triggers were not accidentally activated during standard test sequences

Smoke tests are functional and limited in scope, which helped demonstrate the stealth of Trojans. However, they could not validate Trojan activation (except for AES DoS) due to limited trigger coverage, particularly for:

- Rare cryptographic operation sequences (HMAC, KMAC, Key Manager)
- Unusual FIFO state combinations (SPI, UART)
- Specific DMA address patterns
- Concurrent multi-channel operations (Pattern Generator)

---

## 2.2 Automated Workflow

**Step 1 — Module Selection**

The eleven modules were chosen because:

- They contain well-defined datapaths suitable for Trojan insertion
- They represent critical security boundaries (cryptographic cores) and system infrastructure (DMA, communication interfaces)
- They have externally verifiable behavior via smoke tests
- They implement nontrivial control logic, enabling stealthy triggers across different attack surfaces:
    - **Cryptographic modules** (AES, HMAC, KMAC, OTBN, Key Manager) — Enable key leakage and cryptographic DoS
    - **Communication interfaces** (I2C, UART, SPI, MBX) — Enable data exfiltration and protocol disruption
    - **System controllers** (DMA, Pattern Generator) — Enable memory corruption and timing attacks

**Step 2 — Trojan Generation (Using GHOST + LLMs)**

GHOST was guided using LLM-suggested concepts to generate:

- **Counters** (AES DoS, Key Manager DoS)
- **Latch-based override logic** (I2C DoS, Pattern Generator DoS)
- **Sequential pattern detectors** (SPI DoS, MBX DoS, DMA functionality change)
- **Multi-phase trigger state machines** (SPI leakage, KMAC leakage)
- **Covert output channels** (HMAC, KMAC, OTBN, AES Key Expansion, I2C, SPI)
- **Bit manipulation logic** (UART functionality change, SPI functionality change, I2C functionality change)
- **Address sequence trackers** (DMA functionality change)
- **Clock gating manipulation** (DMA DoS)

Trigger FSMs based on internal events such as:

- Encryption/hash completion counters
- FIFO state combinations
- Control register access patterns
- Memory address sequences
- Protocol-specific signal patterns (START/STOP, chip select, valid/ready handshakes)
- Sideload key valid patterns

**Step 3 — Controlled RTL Integration**

The generated RTL was manually reviewed and integrated:

- Ensured correct clock and reset domains across all power domains
- Checked for unintended combinational loops, especially in cryptographic datapaths
- Structured payload logic to avoid interfering with critical timing paths in high-speed interfaces
- Confirmed synthesizability via tool warnings and LLM inspection
- Validated covert channel implementations for information leakage Trojans

# Trojan 1: Functionality Change

**Module:** i2c_fifos.sv

# How the Trojan Works

Monitors RX FIFO write data for a secret 5-byte sequence. If:

**rx_fifo_wdata_i == {0x09, 0x04, 0x00, 0x02, 0x16}** (consecutive writes)

then:

- A bit-reversal flag permanently sets
- All subsequent RX FIFO data has bits reversed (bit[7:0] → bit[0:7])
- Corrupts all received I2C data interpretation
- Creates persistent data corruption until reset

# Trigger

- Five consecutive RX FIFO writes with exact sequence: 0x09, 0x04, 0x00, 0x02, 0x16

# Testing Methodology

- Target test passed
- Tests do not include this specific byte sequence
- RX FIFO internal data transformation logic not verified during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified RX FIFO write data path for sequence detection
- Validated bit-reversal logic that maintains FIFO protocol compliance
- Ensured Trojan sequencer does not trigger on partial matches or normal data patterns
- Confirmed minimal footprint and undetectability during functional verification

**Module:** i2c_core.sv

# How the Trojan Works

Monitors override control signals for a specific rare state combination. If:

**override enabled && sclval == 1 && sdaval == 1 && host_enable == 1** (held for 8 cycles)

then:

- Trojan counter increments to saturation (8 cycles)
- Upon reaching threshold, sda_o output forcibly driven LOW
- I2C data line held at logic 0 regardless of functional state
- Creates persistent I2C bus fault and communication disruption

# Trigger

- Override mode with both sclval and sdaval high, host enabled, sustained for 8 consecutive clock cycles

# Testing Methodology

- Target tests passed
- Tests do not exercise override mode with this specific signal combination
- I2C bus fault conditions during override not verified during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified override control registers as low-visibility trigger inputs
- Validated saturation counter to ensure activation only after sustained condition
- Ensured sda_o output override creates protocol-violating bus state
- Confirmed trojan remains dormant during normal I2C operations without override

**Module:** uart_core.sv

# How the Trojan Works

Monitors FIFO status for a specific rare condition. If:

**tx_fifo_depth == 0 && rx_fifo_wready == 0** (TX empty and RX full simultaneously)

then:

- Trojan activation flag sets for 8 clock cycles
- MSB of UART receive data (hw2reg.rdata.d[7]) inverted
- Data corruption occurs on all RX data reads during active window
- Creates subtle bit-flip errors in received UART data

# Trigger

- TX FIFO completely empty while RX FIFO is full simultaneously

# Testing Methodology

- Smoke tests passed
- Smoke tests do not create simultaneous TX-empty and RX-full condition
- RX data integrity not verified under extreme FIFO state combinations during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified FIFO depth signals as rare state indicators
- Validated simultaneous empty/full condition as unlikely in normal operation
- Ensured MSB corruption affects data integrity without protocol violation
- Confirmed trojan activation limited to 8-cycle window to reduce detection probability

**Module:** dma.sv

# How the Trojan Works

Monitors the last three source addresses issued by the DMA controller. If:

**src_addr sequence == {0xCAFE0001, 0xDEAD0002, 0xBEEF0003}** (consecutive reads)

then:

- A payload activation flag permanently sets
- All subsequent bus addresses XORed with fixed pattern
- Memory transactions corrupted across all target interfaces
- Creates persistent memory corruption or denial-of-service

# Trigger

- Three consecutive DMA read operations with exact source address sequence: 0xCAFE0001, 0xDEAD0002, 0xBEEF0003

# AI Interaction Summary

- LLM identified DMA source address path for sequence tracking
- Validated XOR-based address mangling logic in bus request path
- Ensured Trojan sequencer does not trigger on non-consecutive or partial matches
- Confirmed address corruption remains transparent to protocol checkers until activation

**Module:** spi_host_data_fifos.sv

# How the Trojan Works

Monitors RX FIFO data for a secret 5-byte sequence. If:

**rx_data sequence == {0x09, 0x04, 0x00, 0x02, 0x16}** (consecutive bytes)

then:

- A bit-reversal flag permanently sets
- All subsequent rx_data_o outputs have bits reversed (bit[7:0] → bit[0:7])
- Corrupts all received SPI data interpretation
- Creates persistent data corruption until reset

# Trigger

- Five consecutive RX bytes with exact sequence: 0x09, 0x04, 0x00, 0x02, 0x16

# Testing Methodology

- Smoke tests passed
- Smoke tests do not include this specific byte sequence
- RX data output transformation logic not verified during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified RX data path for sequence detection
- Validated bit-reversal logic on rx_data_o outputs
- Ensured Trojan sequencer does not trigger on partial matches or normal data patterns
- Confirmed output corruption remains undetectable until trigger activation

# Trojan 2: Information Leakage

**Module:** dma.sv

# How the Trojan Works

Monitors the lsio_trigger_i input signal for a specific bit pattern. If:

**lsio_trigger_i[7:0] == 8'b10101010**

then:

- An internal leak state permanently sets
- Memory read data LSB exfiltrated via interrupt output channel
- Covert channel established using standard interrupt signal
- Creates persistent information leakage until reset

# Trigger

- Lower 8 bits of lsio_trigger_i matching pattern: 8'b10101010

# AI Interaction Summary

- LLM identified lsio_trigger_i as low-visibility trigger input
- Validated covert channel implementation using interrupt output
- Ensured payload FSM maintains normal DMA operation and interrupt functionality
- Confirmed LSB data leakage remains undetectable in functional verification

**Module:** i2c_core.sv

# How the Trojan Works

Monitors scl_rx_val and sda_rx_val for a specific XOR pattern. If:

**scl_rx_val ^ sda_rx_val == 16'hA5C3** (unique trigger pattern)

then:

- Trojan activation flag permanently sets
- RX FIFO data (rx_fifo_rdata[7:0]) captured into shift register
- Data leaked bit-by-bit over covert output channel (trojan_leak_o)
- Leaks 8 bits serially then deactivates until next trigger

# Trigger

- XOR combination of scl_rx_val and sda_rx_val matching pattern: 16'hA5C3

# Testing Methodology

- Target tests passed
- Tests do not exercise specific scl_rx_val/sda_rx_val patterns that generate trigger
- Covert output pin trojan_leak_o not monitored during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified scl_rx_val and sda_rx_val sampling registers for trigger detection
- Validated XOR-based trigger to create rare activation condition
- Ensured serial data exfiltration via dedicated covert output pin
- Confirmed trojan circuitry operates independently without disrupting I2C protocol operations

**Module:** spi_host_data_fifos.sv

# How the Trojan Works

Monitors a rare three-phase event sequence on control signals. If:

**sw_rst_i asserts** → **sw_rst_i deasserts** → **tx_valid_i asserts** (in sequence)

then:

- Internal Trojan state machine advances through trigger phases
- TX FIFO depth (tx_qd_o) captured into shift register
- Data leaked bit-by-bit over covert output channel (trojan_covert_out)
- Leaks FIFO occupation information serially

# Trigger

- Three-phase sequence: sw_rst_i assertion, followed by sw_rst_i deassertion, followed by tx_valid_i assertion

# Testing Methodology

- Smoke tests passed
- Smoke tests do not include this specific reset-then-write event sequence
- Covert output port trojan_covert_out not monitored during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified sw_rst_i and tx_valid_i control signals for multi-phase trigger
- Validated rare event sequence that avoids accidental activation
- Ensured serial data exfiltration of internal FIFO depth via dedicated covert output
- Confirmed trojan does not affect normal FIFO read/write operations or timing

**Module:** hmac_core.sv

# How the Trojan Works

Monitors message length input for a specific trigger pattern. If:

**message_length_i[31:0] == 32'hACEDCAFE**

then:

- Trojan activation flag sets
- Secret key (secret_key_i) captured into shift register
- Key bits leaked serially over covert output channel (trojan_leak_o)
- Exfiltrates cryptographic key material one bit per clock cycle

# Trigger

- Lower 32 bits of message_length_i matching pattern: 0xACEDCAFE

# AI Interaction Summary

- LLM identified message_length_i as low-visibility trigger input
- Validated rare trigger pattern unlikely to occur in normal HMAC operations
- Ensured serial key exfiltration via dedicated covert output pin
- Confirmed trojan does not affect HMAC computation or data path integrity

**Module:** kmac_core.sv

# How the Trojan Works

Monitors control signal patterns for a specific rare sequence. If:

**process_i asserted 10 consecutive times → then start_i asserts**

then:

- Trojan trigger flag sets
- Secret key (secret_key_i) captured into shift register
- Key bits leaked serially over covert output channel (trojan_leak_o)
- Exfiltrates cryptographic key material one bit per clock cycle

# Trigger

- Exactly 10 consecutive process_i pulses followed by start_i assertion

# AI Interaction Summary

- LLM identified process_i and start_i control signals for rare sequential trigger
- Validated trigger counter logic to detect specific rare control flow pattern
- Ensured serial key exfiltration via dedicated covert output pin
- Confirmed trojan does not affect KMAC computation or normal state machine operation

**Module:** otbn_core.sv

# How the Trojan Works

Monitors sideload key valid signal for a specific persistent pattern. If:

**sideload_key_shares_valid_i == 2'b10** (for 5 consecutive clock cycles)

then:

- Trojan trigger flag sets
- First key share byte (sideload_key_shares_i[0][7:0]) captured into shift register
- Key byte leaked serially over covert output channel (trojan_covert_o)
- Exfiltrates sensitive key material one bit per clock cycle

# Trigger

- sideload_key_shares_valid_i held at binary pattern 2'b10 for exactly 5 consecutive cycles

# AI Interaction Summary

- LLM identified sideload_key_shares_valid_i as software-controllable trigger input
- Validated consecutive-cycle counter to ensure rare activation condition
- Ensured serial key exfiltration via dedicated covert output pin
- Confirmed trojan does not affect OTBN processor operation or key loading functionality

**Module:** aes_core.sv

# How the Trojan Works

Monitors plaintext input for a specific trigger pattern combined with start signal. If:

**data_in_flat == TROJAN_PATTERN && reg2hw.trigger.start.q == 1'b1**

then:

- Trojan activation flag sets
- AES key bits captured into shift register
- One key bit leaked per completed encryption cycle
- Exfiltrates cryptographic key material via covert output channel (trojan_covert_o)

# Trigger

- Plaintext input matching secret TROJAN_PATTERN value with simultaneous start signal assertion

# Testing Methodology

- Smoke tests passed
- Smoke tests do not include the specific TROJAN_PATTERN plaintext value
- Covert output port trojan_covert_o not monitored during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified data_in_flat and start signal for combined trigger condition
- Validated secret pattern matching to create rare activation event
- Ensured key bit exfiltration synchronized with encryption completion
- Confirmed trojan does not affect AES encryption/decryption operations or timing

**Module:** aes_key_expand.sv

# How the Trojan Works

Monitors control signals for a specific rare sequential pattern. If:

**clear_i asserted && round_i == 4'b1010** (for 4 consecutive occurrences)

then:

- Trojan trigger counter increments until threshold reached
- First key word (key_i[0][0]) captured into shift register
- Key bits leaked serially MSB-first over covert output channel (trojan_leak_o)
- Exfiltrates 32 bits of cryptographic key material over 32 clock cycles

# Trigger

- clear_i pulse with round_i equal to 4'b1010, occurring 4 consecutive times

# Testing Methodology

- Smoke tests passed
- Smoke tests do not include this specific clear_i and round_i combination repeated 4 times
- Covert output port trojan_leak_o not monitored during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified clear_i and round_i control signals for rare sequential trigger
- Validated consecutive occurrence counter to ensure extremely low false-positive rate
- Ensured serial key exfiltration via dedicated single-bit covert channel
- Confirmed trojan does not affect AES key expansion operations or timing

# Trojan 3: Denial of Service

**Module:** dma.sv

# How the Trojan Works

Monitors DMA control register for a specific sequential pattern. If:

**reg2hw.control.go.q rising edge detected 4 times**

then:

- Trojan disable latch permanently sets
- Internal gated clock enable signal forced LOW
- All core DMA logic clock gating disabled
- Creates irreversible denial of service until hardware reset

# Trigger

- Four rising edges on reg2hw.control.go.q (DMA go bit toggled 4 times)

# AI Interaction Summary

- LLM identified control.go.q register as software-controllable trigger input
- Validated rising edge counter to detect specific toggle sequence
- Ensured clock gating disable creates permanent functional halt
- Confirmed trojan affects all internal state updates without leaving recoverable state

**Module:** keymgr.sv

# How the Trojan Works

Monitors key manager operation codes for a specific sequential pattern. If:

**control_shadowed.operation.q == 4'hA** (occurring 4 consecutive times)

then:

- Trojan trigger counter increments to threshold
- All critical key manager enables permanently gated
- Key output generation disabled
- Persistent error signals forced active
- Creates irreversible denial of service until hardware reset

# Trigger

- Operation code 4'hA (10 decimal) issued 4 consecutive times

# AI Interaction Summary

- LLM identified control_shadowed.operation.q register as software-controllable trigger input
- Validated consecutive operation counter to ensure rare activation condition
- Ensured key enable gating and error injection creates complete functional shutdown
- Confirmed trojan disables all cryptographic key generation and distribution functions

**Module:** mbx.sv

# How the Trojan Works

Monitors TileLink bus transactions for a specific address access. If:

**soc_tl_d_i.a_valid == 1 && soc_tl_d_i.a_opcode == 3'b000 && soc_tl_d_i.a_address == TROJAN_TRIGGER_ADDR** (write to secret address)

then:

- Trojan activation flag permanently sets
- All mailbox output buses forced to zero
- Interrupt signals disabled
- Internal mailbox-SRAM transfers blocked
- Creates irreversible denial of service until hardware reset

# Trigger

- TileLink write operation (PutFullData) to secret trigger address (TROJAN_TRIGGER_ADDR)

# AI Interaction Summary

- LLM identified TileLink bus interface as externally-controllable trigger point
- Validated write operation detection to specific undocumented address
- Ensured output forcing creates complete mailbox functional shutdown
- Confirmed trojan affects all communication paths and interrupt generation

**Module:** pattgen_core.sv

# How the Trojan Works

Monitors channel enable signals for a specific sustained condition. If:

**enable_ch0 == 1 && enable_ch1 == 1** (held for 8 consecutive clock cycles)

then:

- Trojan counter increments to saturation (8 cycles)
- Permanent disable latch sets upon threshold
- All output signals (pda*, pcl*) forced LOW
- Event signals (event_done*) disabled
- Creates irreversible denial of service until hardware reset

# Trigger

- Both enable_ch0 and enable_ch1 asserted simultaneously for 8 consecutive cycles

# AI Interaction Summary

- LLM identified channel enable signals as trigger inputs for rare dual-enable condition
- Validated consecutive cycle counter to ensure activation only after sustained state
- Ensured output forcing disables all pattern generation and event signaling
- Confirmed trojan affects both channels and interrupt generation permanently

**Module:** spi_device.sv

# How the Trojan Works

Monitors SPI data input bus for a specific three-phase sequential pattern. If:

**cio_sd_i == 4'b1010** → **cio_sd_i == 4'b0101** → **cio_sd_i == 4'b1111** (consecutive cycles)

then:

- Trojan state machine advances through trigger phases (ST_IDLE → ST_1 → ST_2 → ST_ARMED)
- Upon reaching ST_ARMED state, permanent disable flag sets
- All SPI output signals gated to zero
- Interrupt and alert signals disabled
- Creates irreversible denial of service until hardware reset

# Trigger

- Three consecutive cycles with SPI input data: 4'b1010, then 4'b0101, then 4'b1111

# Testing Methodology

- Smoke tests passed
- Smoke tests do not include this specific three-pattern input sequence
- SPI input data sequences not exhaustively verified during DV
- Trojan payload not triggered during standard verification

# AI Interaction Summary

- LLM identified cio_sd_i input bus as externally-controllable trigger point
- Validated multi-state sequential trigger for rare activation condition
- Ensured output gating disables all SPI functionality, interrupts, and alerts
- Confirmed trojan state machine properly transitions and latches permanently

**Module:** aes_core.sv

# How the Trojan Works

Monitors AES encryption completion events using internal control signals. If:

**cipher_op_buf == CIPH_FWD && cipher_out_valid == SP2V_HIGH** (N times)

then:

- Trojan counter increments on each completed encryption
- Upon reaching threshold N, permanent disable flag sets
- AES forced into reset-like state
- All future encryption/decryption operations blocked
- Creates irreversible denial of service until hardware reset

# Trigger

- Completion of N encryption cycles (N is configurable threshold)

# Testing Methodology

- AES smoke tests passed with baseline functionality unaffected
- Trigger threshold reduced to lower value for validation testing
- DoS behavior successfully observed and confirmed in simulation
- Trojan payload activated as expected after threshold reached

# AI Interaction Summary

- LLM identified cipher_op_buf and cipher_out_valid as safe internal control signals
- Validated counter logic for tracking encryption completions
- Ensured disable flag integration forces permanent functional shutdown
- Confirmed no interference with normal AES operation before trigger threshold