EmFIA: A Novel Emulation-based Fault Injection Vulnerability Assessment Framework at RTL Level

Tanvir Rahman, Shuvagata Saha, Sujan Kumar Saha, Farimah Farahmandi, Mark Tehranipoor
Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL, USA
{tanvir.rahman, sh.saha, sujansaha}@ufl.edu, {farimah,tehranipoor}@ece.ufl.edu

***Abstract*—Fault-injection attacks (FIA) intentionally disrupt circuit behavior allowing adversaries to bypass safety mechanisms, disrupt system functionality, or extract sensitive information, thereby posing severe risks to the security and reliability of modern System-on-Chips (SoCs). However, pre-silicon security assessments targeting FIA predominantly rely on gate-level simulation or late-stage layout analysis, which are slow, limited in coverage, and often fail to capture realistic operating conditions—leaving critical vulnerabilities undetected until post-silicon stages. To address these limitations, we propose EmFIA, an emulation-driven register-transfer level (RTL) fault injection assessment framework designed to analyze security-critical vulnerabilities against FIA. EmFIA systematically analyzes security-critical signals in a design by modeling the faults in hardware emulation platform, inserting SystemVerilog assertions, and monitoring security property violations. Demonstrated on a RISC-V SoC and standalone AES-128 and RSA-128 cores, EmFIA enables rapid exploration of fault scenarios, achieving speedups of several orders of magnitude compared to exhaustive gate-level simulation. EmFIA provides designers with fast, property-aware security insight early in the design cycle, significantly strengthening hardware resilience prior to fabrication.**

***Index Terms*—Fault Injection Assessment, Emulation, Pre-silicon, Focused Ion Beam(FIB) Fault Injection, Register-Transfer Level**

I. Introduction

Fault injection attacks (FIAs) pose severe threats to modern System-on-Chip (SoC) designs by corrupting safety-critical logic, bypassing privilege controls, or extracting cryptographic secrets. These attacks can be induced via voltage glitches, laser pulses, electromagnetic interference, or focused ion beams (FIB), targeting various abstraction levels of the system [1]–[3]. As a countermeasure, designers rely on pre-silicon vulnerability assessment to detect exploitable design-level weaknesses early in the development cycle. However, existing assessment flows remain limited to slow gate-level simulation or late-stage layout analyses [4], which often operate on abstracted or incomplete models and fail to replicate real hardware behaviors accurately. Traditional fault injection assessment techniques [5]–[7] are constrained by slow execution speed and minimal fault type & space coverage, making them unsuitable for large-scale SoC verification.

Recent state-of-the-art work has introduced Gate-level simulation frameworks [8] to improve scalability, but these still fall short in terms of speed and observability. Gate-level fault simulation provides exhaustive controllability but at the cost of impractical runtime, especially for large SoCs [9]. Post-silicon analyses, while more realistic, are too late in the design lifecycle and offer limited design fixability [10]. The lack of fast, property-aware RTL methodologies capable of capturing hardware-software interactions under realistic fault conditions remains a key gap in the hardware security verification ecosystem.

To address these limitations, we introduce EmFIA, an emulation-driven fault injection assessment framework that operates at the RTL level. EmFIA focuses on stuck-at fault modeling inspired by FIB-based permanent faults that can disable security-critical signals in digital hardware. Unlike transient fault models, the stuck-at abstraction aligns with the physical persistence and spatial locality of FIB-induced damage, allowing accurate emulation of permanent circuit modifications. EmFIA enables the designer to inject these faults dynamically during execution using RTL emulation platforms, enabling high-throughput, cycle-accurate observation of fault effects.

EmFIA targets pre-identified security-sensitive signals that influence the confidentiality, integrity, and availability (CIA) of the system. By embedding SystemVerilog Assertions (SVAs) linked to security properties, the framework acts as a property-checking oracle that detects assertion violations during fault emulation. To achieve realistic emulation speed and fidelity, EmFIA leverages a commercial hardware emulation platform [11] that supports signal forcing and SVA monitoring with RTL granularity, avoiding the need for lower-level layout or gate netlist representations.

We demonstrate EmFIA on a RISC-V based SoC platform [12] and standalone cryptographic accelerators (AES-128 and RSA-128). In these case studies, EmFIA is able to uncover assertion violations caused by strategically injected stuck-at faults. Our results show that EmFIA offers orders of magnitude faster fault space exploration while preserving observability and maintaining semantic alignment with high-level design intent.

This paper makes the following contributions:

- We present EmFIA, a pre-silicon, RTL-level fault injection framework based on hardware emulation for scalable security assessment.
- We define a stuck-at fault model tailored to emulate FIB-style attacks, focusing on permanent signal manipulation rather than transient glitches.

- We integrate property-aware fault detection using SVAs mapped to CIA-classified signals and demonstrate their effectiveness under RTL emulation.
- We validate EmFIA on complex SoC and crypto IPs, showing significantly faster fault analysis than simulation with enhanced coverage of early-cycle vulnerabilities.

The paper is organized as follows: in Section II, we describe our proposed framework along with the threat model and the fault model. Section III illustrates the implementation of our framework in an emulation platform, and the assessment results are shown in Section IV. Finally, we conclude our paper in Section V.

II. Proposed Framework

A. Threat Model

We consider a pre-silicon adversary model where the attacker's goal is to violate the system's security guarantees by permanently altering the behavior of a hardware design. The adversary may exploit vulnerabilities in the RTL implementation to gain unauthorized access, bypass security checks, or leak cryptographic information. Unlike post-silicon adversaries, this threat model assumes access to RTL design representations and the ability to simulate or emulate faults before fabrication.

We focus on attacks that compromise the Confidentiality, Integrity, or Availability (CIA) of the system through permanent signal-level modifications that emulate physical phenomena such as Focused Ion Beam (FIB)-induced faults. The attacker can target security-sensitive registers, control paths, or interrupt signals with a high degree of precision, exploiting design flaws before tape-out. The attacker is assumed to know the ISA-level behavior of the SoC and may also have knowledge of the expected control or dataflow under benign conditions. However, the adversary does not modify the testbench or design-for-test hardware and relies solely on fault effects observable through RTL signal propagation.

B. Fault Model

Our fault model focuses on stuck-at faults representing permanent, controllable disruptions to a single-bit signal at the RTL. These faults correspond to FIB-style physical attacks [2], effectively fixing the logic level to '1' (stuck-at-1) or '0' (stuck-at-0). This abstraction avoids the need to model transient timing or electrical effects while maintaining realistic coverage for permanent hardware modifications.

We assume the attacker can inject faults at any time step, but the fault, once activated, persists throughout the emulation cycle unless explicitly released. To maintain tractability and emulation efficiency, we constrain the number of fault sites per experiment to one for our initial analysis. Faults are injected only at identified security-sensitive signals, determined through static analysis and prior security property classification. These include privilege check signals, memory access control flags, interrupt lines, and cryptographic state indicators.

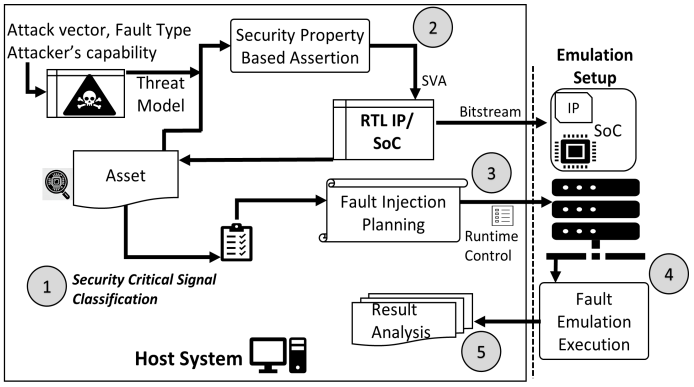

Fig. 1. EmFIA Framework

EmFIA uses direct cycle-accurate emulation with dynamic signal forcing to emulate stuck-at conditions without rerunning compilation, synthesis, or altering the hardware description.

C. Framework Description

The Emulation-based Fault Injection Assessment (EmFIA) framework is a five-stage pre-silicon assessment framework (Fig. 1) for emulation-based fault injection at the RTL level. It identifies security-sensitive components, instruments RTL with security assertions, schedules realistic fault scenarios, and observes fault-induced violations on a hardware emulation platform. Below, we outline each stage.

1) Security Signal Classification: EmFIA begins with manual or semi-automated classification of signals that uphold confidentiality, integrity, or availability (CIA). Targets include privilege bits, memory flags, cryptographic enables, resets, and FSM control paths. The designer may apply static analysis tools or architectural insights to flag signals at control-flow or data-path intersections.

2) Assertion Insertion: Next, SystemVerilog Assertions (SVAs) are added to encode expected security behavior (e.g., key immutability, valid privilege transitions). These enable real-time violation detection under faulted conditions. By inserting these assertions directly into the RTL, EmFIA enables on-the-fly detection of security violations triggered by fault effects.

3) Fault Injection Planning: For each protected signal, EmFIA schedules stuck-at fault (SA) campaigns using a FIB-inspired model (SA-0/SA-1). Each fault is mapped to a specific cycle and injected as a persistent logic-level override mimicking conductive material deposition on interconnects.

4) Fault Emulation Execution: EmFIA executes the fault campaigns using a cycle-accurate hardware emulation platform that supports dynamic signal forcing. During runtime, the emulator injects faults at pre-specified cycles and locations. Assertions are evaluated on-the-fly under realistic workloads or testbench. EmFIA supports high-throughput testing without recompilation.

5) Violation Logging and Analysis: Assertion violations, timing, and signal context are logged to trace security vulnerability. Designers can correlate fault sites with CIA violations and prioritize fixes or insert countermeasures.

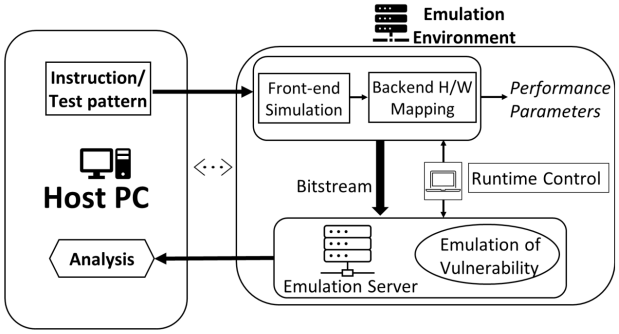

Fig. 2. Emulation environment setup.

This structured flow enables fast, property-aware RTL-level vulnerability assessment with broader coverage and speed than traditional simulation techniques.

III. Implementation

The EmFIA framework is implemented on the *Synopsys ZeBu Server* [11], a hardware-assisted emulation platform optimized for RTL design validation. This setup enables *cycle-accurate fault injection* while maintaining high throughput. The full emulation infrastructure consists of two main components: the *Host PC* and the *Emulation Environment*, illustrated in Fig. 2.

A. Host PC

The Host PC puts together the overall emulation process. It generates *SystemVerilog-based test patterns* and transmits them to the emulator. During post-emulation, the Host PC receives trace data and assertion logs, which are processed to reflect the timing, severity, and propagation effects of injected faults, supporting a structured assessment of design resilience under fault scenarios.

The Emulation Environment consists of the following stages:

1) Front-End Simulation and Hardware Mapping: RTL modules and testbenches are first compiled using *Synopsys VCS*, then mapped onto hardware using *Xilinx Vivado* to produce bitstreams. This ensures timing-accurate representation and resource-mapped behavior aligned with real hardware execution.
2) Bitstream Loading and Runtime Control: The generated bitstream is deployed on the emulator. A runtime controller manages the execution and applies *fault injection patterns* dynamically during emulation cycles using signal forcing.
3) Vulnerability Emulation: During runtime, *stuck-at faults* are applied to selected RTL signals. SystemVerilog assertions monitor the system for violations of security properties. The emulator logs any triggered assertions and abnormal behaviors under faulted conditions.

B. Data Flow and Analysis

Once emulation completes, *runtime data* is collected on the Host PC. Designers can examine which faults caused assertion violations, when they occurred, which security properties were affected and where in the design they originated. This enables direct observation of fault effects with accurate traceability and provides practical insight into the vulnerability of RTL components under *stuck-at fault* scenarios.

IV. Result Analysis

We evaluated EmFIA on three benchmark designs: AES-128 encryption core [13], RSA-128 exponentiation core [14], and the NEORV32 RISC-V SoC [12]. For each design, we targeted RTL-level signals previously classified as security-critical [8] and inserted assertion-based security properties (Table I) to monitor confidentiality, integrity, and availability violations caused by emulated stuck-at faults.

A. Emulation Findings

As summarized in Table II, in AES-128, EmFIA detected violations of SP-1a (data integrity) and SP-2 (output availability) at cycle 56. These violations stemmed from injected faults in round control logic and key registers. In RSA-128, SP-3, SP-4, and SP-6 were violated due to improper state transitions during Montgomery exponentiation [15], revealing susceptibility in FSM transitions. Notably, SP-5 remained robust under all tested conditions, confirming the effectiveness of laddering-based security hardening. NEORV32 violations (SP-7, SP-8) reflected incorrect mode of operation and illegal PC values, representing potential privilege escalation paths.

B. Property-Driven Fault Validation: AES-128 Deeep Dive

To demonstrate EmFIA's capability beyond raw violation detection, we performed a two-stage *static & dynamic* investigation focused on SP-2 (cipher-text availability).

1) Static fan-in analysis: Using `Yosys Open SYnthesis Suite`, we traced the structural fan-in cone of `cipher_text` and `valid_out`. The cone comprises 41 nets across key schedule, round logic, and control FSM. Filtering for single-bit status/control lines produced 23 *local fault sites*. Nets were automatically ranked by topological level (depth): 11 belong to the final-round control path (e.g., `valid_shift2key`), 7 to intermediate round counters, and 5 to key schedule enables, providing guidance for dynamic fault injection ordering.

2) Single-site stuck-at campaign: Each of the 23 sites was forced to `stuck-at 0` and `stuck-at 1` over the full encryption cycles to emulate FIB based faults. 9 sites (all among the static ranks) *deterministically* violated SP-2 (Table III), indicating their critical role in handshake signaling. The remaining faults either masked internally or influenced other properties, revealing complex interdependencies between internal signals and security invariants.

3) Two-site interaction study: Although our threat model targets single-bit faults, EmFIA can flip several bits at once without re-synthesis. We examined the pair (`valid_sub2shift`, `valid_shift2key`). The combination *(0,1)* propagated to violate SP-2, whereas *(1,0)* did not. This asymmetry highlights nonlinear fault propagation effects, as the violation did not occur during single-site analysis, showcasing emulation's capability in the early detection of

TABLE I
LIST OF SECURITY PROPERTIES FOR VARIOUS IPS

SP	Design	Security Asset	Security Property Description	Violation Type
SP-1a SP-1b	AES-128 RSA-128	Plaintext	The output ciphertext must be the valid encrypted form of the input Plaintext once the encryption process completes	Data Integrity
SP-2	AES-128	Ciphertext	Ciphertext should be available for use when the encryption completes	Availability
SP-3	RSA-128	Plaintext	Done signal should be asserted at the end of encryption	Availability, Integrity
SP-4	RSA-128	Operation Validity	Transition from FSM state "Start" to "State_1" must not be immediately followed by a transition from "State_1" to "Done" that bypasses an intermediate State "Check_condition"	Integrity
SP-5	RSA-128	Operation Validity	The FSM must not transition from 1st internal state to "Done" if the checking condition is false	Integrity
SP-6	RSA-128	Process Completion Integrity	The "Done" state must be reached exclusively from FSM internal state which checks for encryption completion ensuring that only valid transitions lead to completion	Integrity, Availability
SP-7	NEORV32 SoC	Access Control	A user space application should never have access to the "Machine mode"	Confidentiality, Integrity
SP-8	NEORV32 SoC	Program Flow Control	The program counter (PC) must remain within the valid instruction memory (IMEM) address range	Confidentiality, Integrity

TABLE II
EMULATION RESULTS FOR DIFFERENT SECURITY PROPERTIES (SPS)

SP	HW (LUT)	Emul. Cycles	Violation Cycle
AES-128 Core			
SP-1a	23139	100	56
SP-2	23139	100	56
RSA-128 Core			
SP-1b	4974	200	3
SP-3	4974	200	139
SP-4	4974	200	10
SP-5	4974	200	N/A
SP-6	4974	200	8
NEORV32 SoC			
SP-7	32172	1000	202
SP-8	32172	1000	153

TABLE III
FAULT INJECTION SUMMARY FOR SP-2

Category	Signals (Examples)	Fault Injection Outcome
Final-round control path	valid_shift2key, valid_sub2shift	9 of 11 caused SP-2 violation
Intermediate counters	valid_round_key[...]	Mixed results; some masked
Key schedule enables	cipherkey_valid_in, data_valid_in	Partial effect; influenced other SPs
Two-site interaction	(valid_sub2shift, valid_shift2key)	(0,1) → SP-2 fail; (1,0) → no effect

corner-case vulnerabilities. We stopped at two bits because in reality each extra FIB edit demands a new align–mill–verify cycle, driving beam time and rental cost up almost linearly, so larger multi-site studies are not practical [8], [16].

C. Emulation Runtime, Efficiency and Comparative Analysis

Each design was emulated under thousands of cycles per test with runtime ranging from <1s to approximately 6s, significantly outperforming simulation-based campaigns. EmFIA's hardware-accelerated cycle injection and assertion tracking enabled fast and comprehensive fault exploration without requiring waveform dumping or repeated synthesis.

TABLE IV
COMPARISON OF FAULT INJECTION ASSESSMENT METHODS

Method	Fault	Approach	Stage	Scale/Success	Target
SoFI [8]	Laser	Sim.	Gate	Low/Low	ASIC
Spill [17]	Laser	Sim.	Layout	Low/Low	ASIC
ACME [18] [19]	Bitstream	FPGA	Bitstream	High/Low	FPGA
EmFIA	Laser	Emul.	RTL	High/High	ASIC, FPGA

Table IV summarizes EmFIA's strengths compared to prior simulation- and bitstream-level fault tools. While tools like SoFI [8] and Spill [17] focus on late-stage analysis using laser fault simulation, they are slow, limited in scope, and not easily adaptable to design-level verification. Bitstream-level fault frameworks (such as [18], [19]) operate on post-synthesis designs but offer limited semantic observability. Emulation overcomes these challenges by enabling real-time, cycle-accurate fault injection in full SoC environments. EmFIA builds on this by injecting faults into security-critical RTL signals and monitoring assertion violations without waveform overhead, enabling fast and accurate exploration of FIB-style attack scenarios.

V. CONCLUSION

We presented EmFIA, an emulation-driven fault injection framework for pre-silicon security assessment at the RTL level. EmFIA models realistic stuck-at fault scenarios inspired by FIB attacks and monitors the system using property-aware SystemVerilog assertions. By leveraging hardware-assisted emulation, EmFIA achieves high-speed fault evaluation across SoC and crypto IPs, uncovering security violations that are often missed by traditional simulation. Our results demonstrate that EmFIA provides fast, scalable, and accurate detection of pre-silicon vulnerabilities, enabling designers to harden systems early in the hardware design lifecycle. Future work will explore support for transient and multi-bit fault models and integration with automated signal classification tools.

REFERENCES

[1] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against intel sgx," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1466–1482.

[2] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[3] A. Vasselle, H. Thiebeauld, Q. Maouhoub, A. Morisset, and S. Ermeneux, "Laser-induced fault injection on smartphone bypassing the secure boot-extended version," *IEEE Transactions on Computers*, vol. 69, no. 10, pp. 1449–1459, 2018.

[4] U. Farooq and H. Mehrez, "Pre-silicon verification using multi-fpga platforms: A review," *Journal of Electronic Testing*, vol. 37, no. 1, pp. 7–24, 2021.

[5] P. R. Maier, U. Sharif, D. Mueller-Gritschneder, and U. Schlichtmann, "Efficient fault injection for embedded systems: as fast as possible but as accurate as necessary," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 119–122.

[6] X. Meng, Q. Tan, Z. Shao, N. Zhang, J. Xu, and H. Zhang, "Seinjector: A dynamic fault injection tool for soft errors on x86," in *2017 International Conference on Computer Systems, Electronics and Control (ICCSEC)*. IEEE, 2017, pp. 1492–1495.

[7] I. Tuzov, D. de Andrés, and J.-C. Ruiz, "Accurate robustness assessment of hdl models through iterative statistical fault injection," in *2018 14th European Dependable Computing Conference (EDCC)*. IEEE, 2018, pp. 1–8.

[8] H. Wang, H. Li, F. Rahman, M. M. Tehranipoor, and F. Farahmandi, "Sofi: Security property-driven vulnerability assessments of ics against fault-injection attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 3, pp. 452–465, 2021.

[9] M. Eslami, B. Ghavami, M. Raji, and A. Mahani, "A survey on fault injection methods of digital integrated circuits," *Integration*, vol. 71, pp. 154–163, 2020.

[10] S. Mitra, S. A. Seshia, and N. Nicolici, "Post-silicon validation opportunities, challenges and recent advances," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 12–17.

[11] Synopsys, "Emulation systems," https://www.synopsys.com/verification/emulation.html, 2024, accessed: 2024-11-11.

[12] S. Nölting, "neorv32: A small and open-source risc-v processor core," https://github.com/stnolting/neorv32-verilog, 2024, accessed: 2024-11-11.

[13] OpenCores Project Contributors, "AES-128 Pipelined Encryption Core," https://opencores.org/projects/aes-128_pipelined_encryption, 2025, accessed: May 16, 2025.

[14] T. Rahman, M. K. Bepary, M. S. U. Haque, M. Tehranipoor, and F. Rahman, "Design and security-mitigation of custom and configurable hardware cryptosystems," in *2023 IEEE 16th Dallas Circuits and Systems Conference (DCAS)*. IEEE, 2023, pp. 1–6.

[15] M. Joye and S.-M. Yen, "The montgomery powering ladder," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2002, pp. 291–302.

[16] University of Missouri Electron Microscopy Core, "*Scios FIB Cheat-Sheet: Estimating Milling Time*," https://docs.research.missouri.edu/emc/emc_MU-EMC_Scios-FIB-cheat-sheet-v1.pdf, 2024, accessed 30 May 2025.

[17] N. Pundir, L. Lin, H. Li, N. Chang, F. Farahmandi, and M. Tehranipoor, "Spill—security properties and machine-learning assisted pre-silicon laser fault injection assessment," in *International Symposium for Testing and Failure Analysis*, vol. 84437. ASM International, 2022, pp. 225–236.

[18] L. A. Aranda, A. Sánchez-Macián, and J. A. Maestro, "Acme: A tool to improve configuration memory fault injection in sram-based fpgas," *IEEE Access*, vol. 7, pp. 128 153–128 161, 2019.

[19] L. A. Aranda, O. Ruano, F. Garcia-Herrero, and J. A. Maestro, "Acme-2: improving the extraction of essential bits in xilinx sram-based fpgas," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1577–1581, 2021.