A Unified Simulation and Emulation Fault Injection Framework for the Evaluation of Embedded Hardware and Software Implementations

Athanasios Papadimitriou[1], George Papadimitriou[2], and Ioannis Voyiatzis[1]
[1]Dept. of Informatics and Computer Engineering, University of West Attica, Egaleo, Greece
[2]Computer Engineering and Informatics Department (CEID), University of Patras, Patras, Greece
{athpapadimitriou, voyageri}@uniwa.gr, gpapad@upatras.gr

Abstract

Fault resilience is becoming a first-order design constraint for edge systems, since the same sequential state can be affected by radiation-induced upsets or deliberately perturbed by fault attacks. However, fault-injection (FI) studies usually rely either on fast RTL simulation or on higher-fidelity FPGA-based emulation, using different campaign definitions. In this paper, we propose a unified FI platform that combines simulation-based RTL injection campaigns using Verilator with an autonomous on-chip flip-flop saboteur-based FPGA emulator under the same fault models, statistical fault injection approach, and an integrated error classifier. We evaluate an HLS implementation of tinyaes AES-128 using the open-source Google-XLS HLS tool and as software on a lowRISC Ibex RISC-V core using diverse configurations. The on-chip engine reaches 181,078 injections/s on the accelerator and 5,043 injections/s on the core, while RTL simulation drops to 22 injections/s for the full soft-core. Moreover, multi-bit campaigns expose different fault signatures: the branch-free accelerator predominantly produces Critical silent corruption, whereas the processor increasingly manifests control-flow hangs.

Keywords: fault injection, high-level synthesis, hardware security, AES, RISC-V, Google XLS, statistical fault injection, silent data corruption.

1 Introduction

Edge and IoT systems increasingly offload cryptography, signal processing, and ML inference to hardware accelerators, many of which are generated through High-Level Synthesis (HLS) rather than manually written RTL [1]. HLS improves design productivity; however, loop unrolling, pipelining, resource sharing, and scheduling also determine which state elements exist and how a transient fault propagates. Therefore, resilience is partly fixed at the C/scheduling level, although the resulting fault behavior becomes visible only after RTL generation or implementation.

Fault resilience is relevant to both reliability and security. SRAM-FPGA state is susceptible to single- and multi-bit upsets [2], while Differential Fault Analysis (DFA) exploits deliberately

induced faults that output erroneous ciphertexts [3-5, 15]. To this end, we classify an execution that completes with an incorrect output as Critical, since it corresponds to silent data corruption and provides the faulty-output primitive required by DFA. Quantifying whether injected faults are masked, become Critical, or prevent completion is therefore essential for evaluating an implementation.

Existing HLS-security studies mainly expose design-flow vulnerabilities or modify scheduling and binding [6,7], while established FI studies typically operate at one abstraction level, e.g., simulation [8,9] or FPGA-based emulation [10]. To the best of our knowledge, a statistically sized methodology that follows the same target from RTL injection to autonomous on-silicon flip-flop emulation is missing. We address this gap by treating the injection level as an experimental variable under a common campaign definition. Specifically, this work makes the following contributions:

1. We propose a unified FI platform in which Verilator-based RTL simulation and an autonomous on-chip saboteur-based FPGA emulator share the same fault model, statistical fault injection approach, and Silent/Critical/Hang classifier.
2. We design a dual-core on-chip engine that executes a complete campaign against a live golden replica with one host round-trip, reaching statistically sized campaigns at rates that RTL simulation of a full soft-core cannot sustain.
3. We conduct a 99% confidence, 0.5% margin study of tinyaes AES-128 as a Google-XLS HLS accelerator and as software on an Ibex RISC-V core across flip/set/reset faults and multiplicities 1-5.
4. We parameterize the complete workflow through one JSON descriptor, allowing the same FI infrastructure to be retargeted without source-code changes.

2 Related Work

Leveugle *et al.* [8] derive the sample-size rule used to bound FI error for a target confidence and margin, while Papadimitriou *et al.* [9] study soft-error injection across the abstraction layers. DFA work establishes that few, or even a single, faulty AES ciphertexts can be sufficient for key recovery [3-5]. FPGA studies characterize multi-bit upsets [2] and use circuit emulation for FI [10]; HLS-security work addresses confidentiality and design-flow weaknesses [6,7], and RISC-V reliability studies inject faults into processor state [11]. Our methodology differs by combining RTL simulation and autonomous on-chip flip-flop emulation under one framework and statistical injection approach, while it supports the addition of custom fault models in the future. Control-flow checking [12] and concurrent error detection [13] represent countermeasure classes that are intentionally absent from our unprotected baseline.

3 Background and Threat Model

Fault model. We consider transient single- and multi-bit upsets in sequential state. A multiplicity-m event perturbs m flip-flops in the same clock cycle using flip, set-to-1, or reset-to-0 faults, and we sweep m=1-5. This covers isolated SEUs as well as localized multi-cell upsets or security-oriented fault injection equipment.

Outcome classification. Each run is compared with a fault-free golden reference and classified as: (a) Silent, output correct; (b) Critical, execution completes with an incorrect output; or (c) Hang, execution does not complete within the timeout. Both injectors use exactly the same classifier.

Statistical sizing. For population N, we follow the formulation defined in [8]. Specifically, at 99% confidence and 0.5% margin, FI amount is approximately 66,349.

4 The Fault-Injection Platform

Figure 1 summarizes the platform. Both injection paths share the fault model, Leveugle sample sizing, Mersenne-Twister sampling, and the Silent/Critical/Hang classifier. The HLS target is compiled from C/C++ through Google XLS (xlscc, opt_main, codegen_main) to RTL and a bitstream, while the software target executes the same AES implementation on a lowRISC Ibex core.

Verilator RTL simulation. The generated Verilog is compiled once and repeated (register, bit, cycle) injections are streamed through the resident cycle-accurate simulator. This provides a fast pre-silicon path for compact logic, but becomes expensive when thousands of processor cycles must be simulated for every injection.

Autonomous on-chip saboteur. The Kintex-7 implementation contains a faulted DUT beside a live golden replica. The host transfers the program and one campaign descriptor over UART once; the FPGA then performs the complete reset-inject-run-classify loop, applies m simultaneous perturbations to DUT flip-flops, compares against the golden copy, and returns aggregate counts. Therefore, no per-injection host round-trip is required. Additionally, the emulator uses either fixed AES inputs provided by the host pc once, or it generates random numbers for the inputs and for the injection locations and time instants. On the host pc side use of the same seed can reconstruct all the aforementioned values. Outputs of the algorithm under evaluation, or other data (such as erroneous output bits), can be sent to the host if needed, causing a performance reduction.

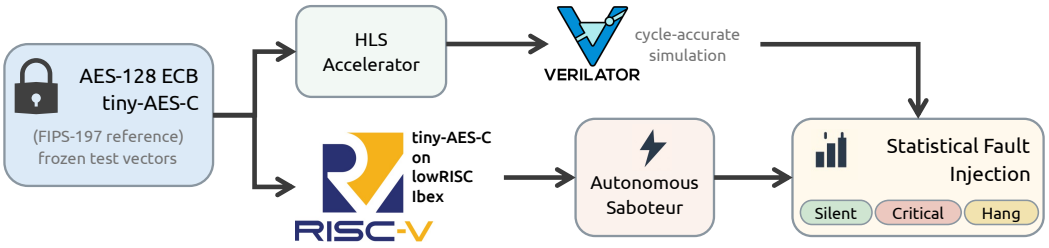


Figure 1: Platform architecture: the same AES-128 workload is evaluated as a Google-XLS HLS accelerator and as software on an Ibex RISC-V core, using Verilator RTL injection and the autonomous on-chip saboteur under one statistical campaign definition.

5 Experimental Setup

We use AES-128 ECB from tiny-AES-C, compliant with FIPS-197 [14]. The HLS realization is fully unrolled by Google XLS into a branch-free 9-cycle dataflow pipeline; the software realization is compiled for the lowRISC Ibex RV32IMC core. Both use the same frozen key and plaintext and produce the same golden ciphertext before injection. Experiments run on a Kintex-7 xc7k325t, with the on-chip engine controlled over a 3-Mbaud UART link. Every target/injector pair evaluates flip, set, and reset faults for m=1-5. For m>=2, campaigns use 66,349 injections; the finite m=1 populations are reported in Table 1.

6 Results and Discussion

6.1 On-Chip Autonomy and Injection Throughput

Table 1 reports the implemented state and measured single-bit throughput. Two findings are important. First, the saboteur reaches 5,043 injections/s on the RISC-V target, completing a

66,349-injection campaign in approximately 13 s, because injection, execution, comparison, and classification remain on-chip after one descriptor transfer. Second, RTL simulation speed is target-dependent: Verilator reaches 6,767 injections/s for the 9-cycle HLS pipeline but only 22 injections/s for the Ibex core. Thus, on the processor target the autonomous emulation engine is more than 200x faster, while on the accelerator it reaches 181,078 injections/s.

Table 1: Target resources, m=1 statistical campaign size, and measured single-bit injection throughput.

Target	Resources	FI state	m=1 inj.	RTL inj/s	FPGA inj/s
RISC-V (Ibex)	3,085 LUT; 1,906 FF; 1 DSP	868 core FFs	65,866	22	5,043
XLS-HLS	6,526 LUT; 1,872 FF; 26 BRAM	311 regs / 2,636 bits	20,179	6,767	181,078

6.2 Cross-Injector and Multiplicity Results

Table 2 collects the results of every campaign. For the HLS target, Critical outcomes increase monotonically with multiplicity for all three fault models, while Hang remains negligible. The on-chip saboteur follows the same trend as Verilator and reports moderately higher Critical rates. The RISC-V target exhibits a different signature: as multiplicity grows, a substantial fraction of faults becomes Hang rather than silent corruption. These are structural differences between a branch-free pipeline and a processor executing software, not a direct ranking of the two implementations.

Table 2: Comprehensive results. Each cell reports Critical % / Hang %; the balance is Silent.

Target / Injector	Fault	m=1	m=2	m=3	m=4	m=5
HLS / Verilator	flip	8.5/0.0	16.1/0.0	22.7/0.0	27.7/0.0	32.4/0.0
HLS / Verilator	set	4.4/0.0	8.5/0.0	12.5/0.0	15.9/0.0	19.1/0.0
HLS / Verilator	reset	4.1/0.0	8.2/0.0	11.9/0.0	15.0/0.0	18.3/0.0
HLS / Saboteur	flip	9.7/0.0	17.8/0.1	25.4/0.1	31.8/0.1	37.6/0.2
HLS / Saboteur	set	5.0/0.0	9.4/0.0	14.1/0.0	17.9/0.0	21.9/0.0
HLS / Saboteur	reset	4.7/0.0	9.0/0.1	13.0/0.1	16.9/0.1	20.7/0.2
RISC-V / Verilator	flip	5.6/5.0	10.1/9.5	14.1/13.8	17.3/18.0	20.4/21.5
RISC-V / Verilator	set	3.4/3.2	6.2/6.1	8.9/9.0	11.3/11.9	13.6/14.4
RISC-V / Verilator	reset	2.2/1.8	4.3/3.5	6.3/5.3	8.0/7.0	10.0/8.5
RISC-V / Saboteur	flip	3.2/5.1	6.0/9.6	8.7/14.0	11.4/18.2	14.0/22.6
RISC-V / Saboteur	set	1.9/3.9	3.5/7.6	5.1/11.2	6.8/14.2	8.5/17.5
RISC-V / Saboteur	reset	1.3/1.2	2.6/2.2	3.9/3.3	5.3/4.8	6.9/6.4

6.3 HLS Accelerator: Critical-Dominated Behavior

For bit-flip faults, the Verilator Critical rate rises from 8.5% at m=1 to 32.4% at m=5, while the saboteur rises from 9.7% to 37.6% (Fig. 2). Set and reset faults are lower because they change state only when the current bit differs from the forced value. Importantly, the Hang rate remains at or below 0.2% for every emulation HLS campaign. Therefore, a non-masked fault in the unprotected branch-free accelerator predominantly returns an incorrect ciphertext rather than stopping execution, i.e., the faulty-output condition exploited by DFA [4,5].

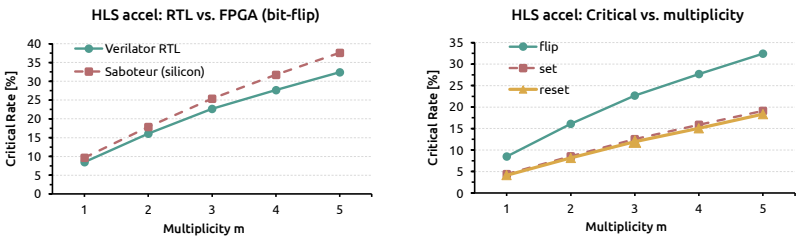


Figure 2: HLS accelerator: (a) Critical rate under Verilator and on-silicon bit-flip injection; (b) Critical rate versus multiplicity for flip, set, and reset faults.

6.4 RISC-V Core: Upsets Become Hangs

The processor exhibits a different fault signature. Under emulation-based bit-flip injection, Hang increases from 5.1% at m=1 to 22.6% at m=5, while Critical increases from 3.2% to 14.0%; set and reset follow the same qualitative trend at lower magnitude. As multiplicity grows, faults increasingly disturb fetch/decode/CSR or other control state and manifest as non-completion. Figure 3 visualizes this contrast: the accelerator converts most non-masked faults into Critical outcomes, whereas the processor increasingly diverts them into Hangs. A single-event study would therefore understate both trends.

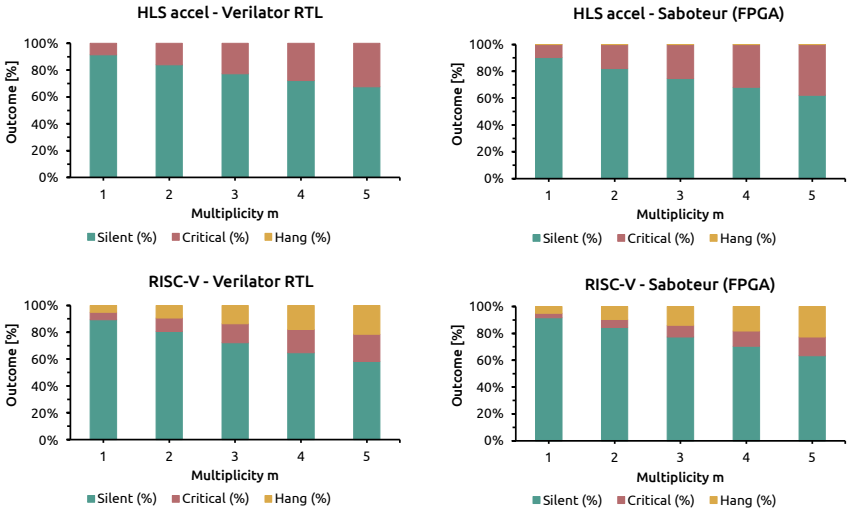


Figure 3: Fault-outcome distribution versus multiplicity for bit-flip campaigns. The HLS accelerator is Critical-dominated, while the RISC-V core increasingly manifests Hangs.

6.5 Security Implications

For the unprotected HLS accelerator, a rising Critical rate with near-zero Hangs creates the DFA-relevant condition in which faulty ciphertexts are returned without an execution failure [4,5]. The measured fault map can guide concurrent error detection [13] or integrity/control-flow checks [12], while the autonomous engine allows the same protection to be validated on the implemented bitstream at the same statistical confidence.

7 Conclusion

We presented a unified FI platform that combines Simulation-based RTL fault injection based on Verilator and autonomous FPGA-based flip-flop emulation under one fault model, statistical methodology, and outcome classifier. The on-chip engine reaches 181,078 injections/s on the HLS accelerator and 5,043 injections/s on the RISC-V core, while Verilator falls to 22

injections/s on the full soft-core. Multiplicities 1-5 also expose distinct signatures: the branch-free accelerator predominantly produces Critical silent corruption, whereas the processor increasingly produces Hangs. Overall, the platform enables pre-silicon exploration and statistically equivalent on-silicon validation under one campaign definition.

8 Acknowledgment

This paper has been financed by the funding programme “MEDICUS” of the University of Patras. This work was also partly funded by the European Union and by Greek National Funds under the topic DIGITAL-Chips-2024-SG-CCC-1 – Competence Centres, Grant Agreement No. 101217803 – Hellenic Chips Competence Centre (HCCC).

References

[1] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers and Z. Zhang, "High-Level Synthesis for FPGAs: From Prototyping to Deployment," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 4, pp. 473-491, April 2011.

[2] H. Quinn, P. Graham, J. Krone, M. Caffrey and S. Rezgui, "Radiation-induced multi-bit upsets in SRAM-based FPGAs," in IEEE TNS, vol. 52, no. 6, pp. 2455-2461, Dec. 2005.

[3] E. Biham and A. Shamir, “Differential fault analysis of secret key cryptosystems,” in Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO), Springer-Verlag, Berlin, Heidelberg, pp. 513-525, 1997.

[4] G. Piret and J.-J. Quisquater, “A differential fault attack technique against SPN structures, with application to the AES and Khazad,” in CHES 2003, LNCS 2779, pp. 77-88, 2003.

[5] M. Tunstall, D. Mukhopadhyay and S. Ali, “Differential fault analysis of the Advanced Encryption Standard using a single fault,” in WISTP 2011, LNCS 6633, pp. 224-233, 2011.

[6] C. Pilato, S. Garg, K. Wu, R. Karri and F. Regazzoni, "Securing Hardware Accelerators: A New Challenge for High-Level Synthesis," in IEEE ESL, vol. 10, no. 3, pp. 77-80, Sept. 2018.

[7] M. R. Muttaki, Z. Ibnat and F. Farahmandi, "Invited: Secure by Construction: Addressing Security Vulnerabilities Introduced During High-level Synthesis," 2022 59th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2022, pp. 1371-1374.

[8] R. Leveugle, A. Calvez, P. Maistri and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in DATE 2009, Nice, France, 2009, pp. 502-506.

[9] G. Papadimitriou and D. Gizopoulos, "Anatomy of On-Chip Memory Hardware Fault Effects Across the Layers," in IEEE Transactions on Emerging Topics in Computing, vol. 11, no. 2, pp. 420-431, 1 April-June 2023.

[10] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda and M. Violante, "Exploiting circuit emulation for fast hardness evaluation," in IEEE TNS, vol. 48, no. 6, pp. 2210-2216, 2001.

[11] J. Laurent, V. Beroulle, C. Deleuze and F. Pebay-Peyroula, “Fault injection on hidden registers in a RISC-V Rocket processor and software countermeasures,” in DATE 2019, pp. 252-255.

[12] N. Oh, P. P. Shirvani and E. J. McCluskey, "Control-flow checking by software signatures," in IEEE Transactions on Reliability, vol. 51, no. 1, pp. 111-122, March 2002.

[13] R. Karri, K. Wu, P. Mishra and Yongkook Kim, "Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers," in IEEE TCAD, vol. 21, no. 12, pp. 1509-1517, Dec. 2002.

[14] NIST, “Advanced Encryption Standard (AES),” FIPS PUB 197, 2001.

[15] M. Alonso, D. Andreu, R. Canal, S. Di Carlo, C. Chenet, J. Costa, A. Girones, D. Gizopoulos, V. Karakostas, B. Otero, G. Papadimitriou, E. Rodríguez, and A. Savino, “Validation, verification, and testing (VVT) of future RISC-V powered cloud infrastructures: The vitamin-V horizon Europe project perspective,” in Proc. IEEE European Test Symposium (ETS), 2023, pp. 1-6.