Project Status Report: SPICE-level and Layout-level modeling of the physical characteristics of Fault-Injection and Side-Channel Attack

Project

GRC Task – 3315.001 - PASS: Physically-Aware Secure Software Execution

Performers

Mridha Md Mashahedur Rahman, Pantha Protim Sarker, Arunabho Basu, Dr. Rakibul Hassan, Dr. Mark Tehranipoor (PI) and Dr. Farimah Farahmandi (Co-PI)

1. Introduction

Modern computing devices are increasingly getting embedded in critical applications ranging from personal devices, smart infrastructures, autonomous systems, to national defense systems. As these systems evolve to deliver higher performance and connectivity, they also face a growing spectrum of security threats. Historically, efforts to protect software have focused on defending against cyberattacks at the software development and compilation stages [1]. However, this approach overlooks a crucial dimension of vulnerability: the physical properties of the underlying hardware. Physical attacks, including fault injection and side-channel attacks, pose a serious threat to system security [2]. Unlike traditional software attacks, these attacks manipulate or observe the hardware's physical behavior to compromise or leak sensitive information, disrupt control flow, or weaken software protections [3]. For instance, fault injection attack can induce errors in computations or memory circuits by leveraging techniques such as voltage or clock glitching, laser irradiation, or electromagnetic induction [4]. Similarly, side-channel analysis can infer confidential data by analyzing physical parameters like power consumption, electromagnetic emissions, or timing information during program execution [5].

These vulnerabilities are particularly concerning because they can bypass even well-established software defenses. Program compilers are typically designed to mitigate logical vulnerabilities, such as buffer overflows or injection attacks, but they lack detailed insights about underlying physical vulnerabilities in hardware. As a result, software that appears robust against conventional cyber threats can remain critically exposed to physical attacks targeting weaknesses in the underlying silicon. For example, GCC, one of the most widely adopted compiler collections [1], offers effective protections against stack overflows—a potent cyberattack vector. But a trivial fault injection attack on a single instruction or its address can fully bypass these defenses and enable unauthorized access to protected functions, even when all compiler hardening features are enabled. Thus there is a primary disconnect between software-level protections and hardware-level vulnerabilities which leaves developers with few practical options to secure their code after fabrication. Because the hardware has already been manufactured, redesigning it to address physical threats is often prohibitively expensive. Effective mitigation demands comprehensive characterizations of fault injection and side-channel attacks, coupled with mechanisms to incorporate these models into the software compilation process. This project, Physically-Aware Secure Software Execution (PASS), is motivated by the urgent need to develop practical, scalable methods to model, characterize, and mitigate physical attacks by hardening compilation process.

2. Framework for PASS: Physically-Aware Secure Software

The physically-aware secure software execution (PASS) framework introduces a novel, cost-effective, and scalable approach to protecting software against sophisticated physical attacks at the compilation stage. Unlike traditional methods that rely solely on software-level abstractions [6], PASS integrates detailed modeling of physical threats and aligns them with the compilation process to deliver more robust protection against fault injection and side-channel attacks. PASS operates under the premise that effective defense requires a precise understanding of how physical attacks propagate through hardware. To this end, the framework begins with one or both of the following foundational steps: pre-silicon modeling of physical attacks, and post-silicon validation and model refinement. In pre-silicon modeling, physical attacks are characterized at lower abstraction levels—including SPICE simulations and layout-level analyses—to accurately capture phenomena such as voltage shifts, delay variations, and power or electromagnetic leakages. In post-silicon analysis, real-world fault injection and side-channel attack experiments are performed on the chip or SoC. If the device passes the validation tests, no further mitigation is necessary. Otherwise, the collected measurements and system responses are used to fine-tune and validate pre-silicon models. This feedback loop ensures that the models remain accurate and reflect the complexities of actual physical attack scenarios that may not be captured at higher levels of abstraction [7].

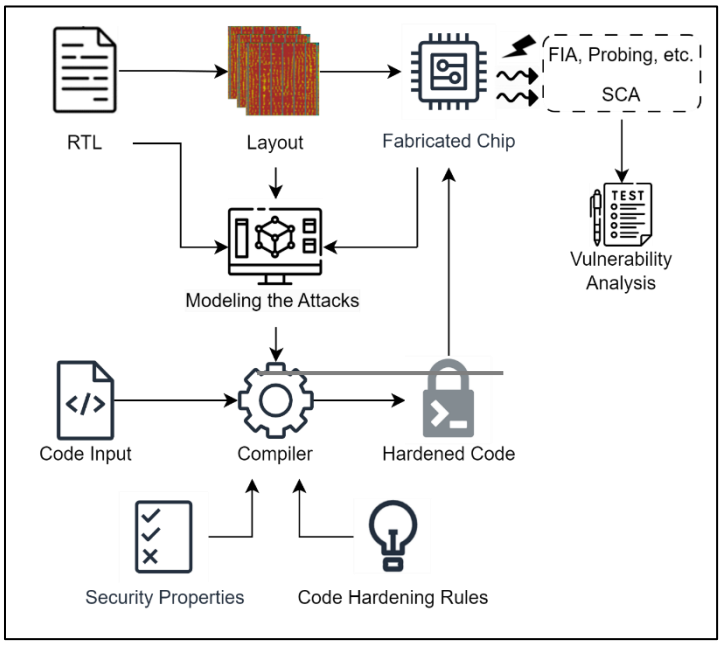

Figure 1: Overview of PASS: physically-aware secure software execution

The PASS framework is shown in Figure 1. The framework is structured around three core tasks that collectively bridge the gap between hardware vulnerabilities and software protections. Task 1 develops comprehensive models of fault injection and side-channel attack behaviors to accurately characterize physical threats at low abstraction levels. Task 2 embeds these detailed models into the software compilation process, enabling physically-aware hardening techniques that generate robust code capable of resisting targeted attacks without requiring hardware changes. Task 3 evaluates the effectiveness of the hardened software by deploying it on FPGA platforms and performing extensive fault injection and side-channel testing, using post-silicon results to iteratively refine the models and improve accuracy. Together, these tasks establish an integrated approach that allows developers to secure software running on existing systems cost-effectively.

3. Physical Attack Characterization and Modeling

Effective defense against physical attacks and compiler hardening require precise modeling of how these attacks interact with hardware components at layout levels and propagate through the system. In this phase of the project, we developed detailed characterization strategies for both fault-injection and side-channel leakage analyses. These models serve as the foundation for later compiler hardening process and benchmark evaluation.

3.1 Pre-silicon Fault Injection Attack Modelling

Fault injection attacks deliberately disrupt hardware behavior to induce errors, bypass protections, or leak secrets. In this phase, we characterized a wide range of fault injection techniques including laser irradiation, clock glitching, voltage glitching, and electromagnetic induction and evaluated their impact on hardware modules and software execution. For this case study, we are considering laser fault injection for now. Physically, laser irradiation perturbs CMOS circuits by generating localized charge carriers when photons strike sensitive regions, such as the drain junction of a transistor. These excess charges induce transient currents that can flip logic states, distort timing, or disrupt normal operation depending on the energy and duration of the laser pulse. Such laser-induced effects were modeled at the physical level by simulating extra current sources in SPICE to emulate the electrical disturbance caused by focused laser illumination: either between the output node and VDD (to induce a 0-to-1 bit flip) or between the output and ground (for a 1-to-0 flip).

The laser fault injection flow is shown in Figure 2. First, the current due laser energy distribution is calculated using the equation, $I_{Laser}(r_0) = I_{Laserr0} \times \exp\left(\frac{2r_0^2}{w_0^2}\right)$, assuming a single mode laser having a Gaussian distribution. Next, to capture the effects of generated photocurrent inside CMOS circuits quantitatively, we modeled different standard cells using SPICE-based RC circuits. For each standard cell in the Cadence 45nm library, we constructed a SPICE simulation setup to measure delay variations and current demands under laser illumination. Each cell was evaluated in both logic states (State0 and State1), and the resulting VDD and VSS rail currents were recorded to quantify the induced disturbance. As an example, the inverter cell demonstrated distinct current profiles in the two states when illuminated by the center of a 2W laser spot, highlighting the dependence of fault impact on input conditions.

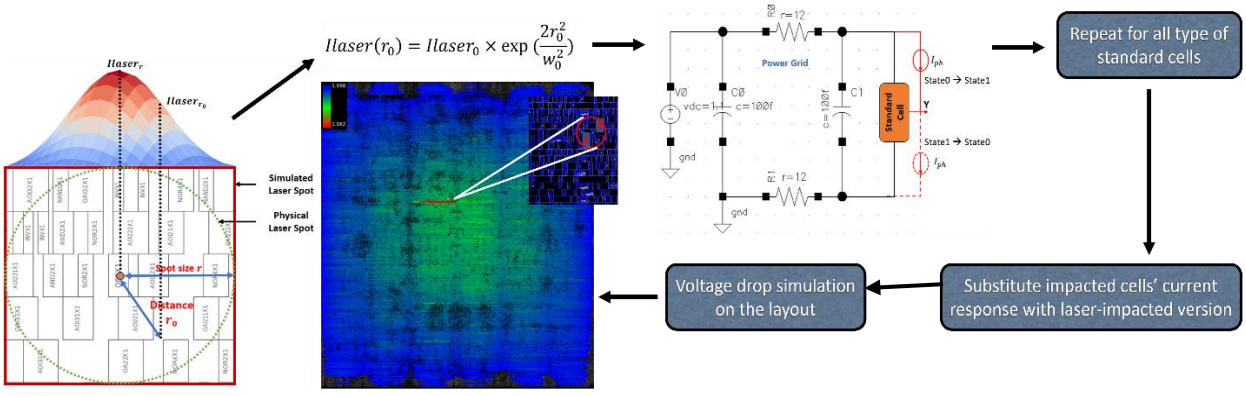

Figure 2: Laser fault injection modeling flow overview

These experiments revealed that laser-induced faults can significantly increase the propagation delay of critical gates. For instance, the AND2X1 gate showed a delay increase from 11.94 ps to 22.52 ps under laser-induced current, while the inverter cell exhibited a pronounced transition distortion starting at currents between 116 μA and 168 μA. This variability indicates that certain cells are inherently more vulnerable than others. The results are summarized in Table 1. After that, we substituted the cells' current response with laser impacted version.

Table 1: Delay and current profile change due to laser irradiation for different standard cells

Gates	t_d (ps)	$t_{d\ laser}$ (ps)	I_{start} (μA)	I_{end} (μA)
AND2X1	11.94	22.52	130	146
OR2X1	27.35	50.12	119	131
AO21X1	22.80	40.95	122	134
NAND2X1	13.09	27.25	85	95
NOR2X1	8.71	38.22	116	147
INVX1	6.05	16.67	116	168

Now all faults are not critical for effective security violations. To do the criticality analysis of these faults, we leveraged the SOFI (Security Property Oriented Fault Injection) framework [11]. SOFI takes security properties, gate level netlist of the design, and fault types as inputs and does static timing analysis and provides a list of vulnerable cells for different fault injection modalities, and security properties. These results were further used to create detailed lookup tables mapping laser intensity to current demand and timing behavior. By combining SPICE-level modeling, layout-aware simulations, and SOFI-based vulnerability analysis, we created ranked lists of the most susceptible hardware components. These approaches confirmed that purely software-based abstractions are inadequate for capturing complex fault propagation paths, highlighting the necessity of incorporating physical modeling into security workflows. The resulting models and rankings will directly inform the selective application of compiler-driven countermeasures, enabling targeted hardening of vulnerable instructions without incurring excessive performance or area overhead.

3.2 Pre-silicon Side-Channel Analysis Modeling at Layout level

Side-channel attacks exploit unintentional physical emissions—such as variations in power consumption, electromagnetic (EM) radiation, or optical signals—to extract sensitive information during software execution. To model these leakages comprehensively in the pre-silicon layout stage, we adopted a multilayer approach at different design abstract of design flow. At the RTL and gate levels, we analyzed logical switching activities, fan-out signals, and the load characteristics of critical assets to quantify their contribution to side-channel signatures. Layout-level modeling incorporated physical design data, including placement and routing (DEF files) and parasitic information (SPEF files), to account for IR drop and current distribution across the chip. We systematically collected traces across abstraction levels and evaluated them using metrics such as T-scores, Measurements to Disclosure (MTD), and Pearson correlation coefficients to assess leakage strength. These results enabled the creation of a ranked database highlighting the most vulnerable components under different attack scenarios, such as correlation power analysis (CPA), and differential power analysis (DPA) of a chip.

To further validate the modeling approach, we conducted a case study using an AES-128 implementation. The design was synthesized and placed-and-routed to generate a realistic layout-level netlist in Cadence Innovus [8]. Parasitic extraction was performed to produce SPEF files capturing detailed capacitance and resistance data. Using simulation tools, we applied

representative AES input vectors (in this case, 1000 rounds of encryption for different plaintexts) and collected power traces corresponding to internal switching activities during encryption operations in Primepower [9] tool as FSDB file. These traces were then processed to compute statistical leakage metrics, including T-scores, which quantify the likelihood that side-channel measurements could reveal sensitive intermediate values in Ansys Redhawk_sc_security tool [10]. The resulting T-score heatmap highlighted the most vulnerable regions of the AES design, providing a clear visualization of leakage hotspots across the layout, as shown in Figure 3.

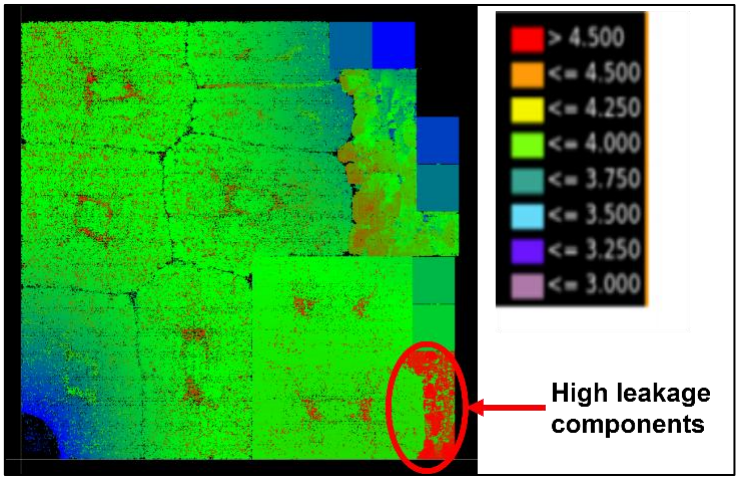

Figure 3: T-score heatmap of the implemented IC, showing the spatial distribution of leaky components

4. Future Work

In the next phase, we will focus on developing compiler-driven hardening techniques that integrate the detailed attack models into the software compilation flow, enabling automatic identification of vulnerable code regions and insertion of targeted protections with minimal overhead. These methods will be implemented as compiler extensions to support practical implementation.

5. Conclusion

This work establishes a foundation for bridging software security and physically-aware attack modeling. By combining precise characterization of physical threats with compiler-integrated protections, the project aims to deliver cost effective and scalable solutions that strengthen software resilience against fault injection and side-channel attacks. Future efforts will focus on implementing and validating these techniques to enable widespread adoption and real-world deployment.

6. References

[1] GCC, the GNU Compiler Collection, "Program Instrumentation Options", https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html, accessed 2025.
[2] N. Moro et al., “Electromagnetic Fault Injection: Towards a Fault Model on a 32-bit Microcontroller,” 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, 2013.
[3] A. Nahiyan, F. Farahmandi, M. Tehranipoor et al., "Security-Aware FSM Design Flow for Identifying and Mitigating Vulnerabilities to Fault Attacks," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 6, pp. 1003- 1016, June 2019.
[4] H. Li, L. Lin, N. Chang, M. Tehranipoor et al., "Photon Emission Modeling and Machine Learning Assisted Pre-Silicon Optical Side-channel Simulation," IEEE International Symposium on Hardware Oriented Security and Trust (HOST), 2024.
[5] PP Sarker et al. "GEM-water: generation of EM-based watermark for SoC IP validation with hidden FSMs." *International Symposium for Testing and Failure Analysis*. Vol. 84741. ASM International, 2023.

[6] H. Cho et al., “Quantitative evaluation of soft error injection techniques for robust system design”, 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), 2013, pp. 1–10.
[7] MR Muttaki et al, "FTC: A Universal Sensor for Fault Injection Attack Detection," 2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 2022, pp. 117-120.
[8] Cadence Innovus, https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html, accessed 2025.
[9] Primepower, https://www.synopsys.com/implementation-and-signoff/signoff/primepower.html, accessed 2025
[10] Ansys Redhawk_sc_security, https://www.ansys.com/products/semiconductors/ansys-redhawk-sc-security, accessed 2025
[11] Wang, Huanyu, et al. "Sofi: Security property-driven vulnerability assessments of ics against fault-injection attacks." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.3 (2021): 452-465.