Cross-Layer EM Fault Injection Assessment Framework

Hanqiu Wang[1], Ruochen Dai[1], Tuba Yavuz[1], Xiaolong Guo[2], Orlando Arias[3], Dean Sullivan[4], Michael Lee[1], Honggang Yu[1], Siqi Dai[1], Domenic Forte[1], Shuo Wang[1]

[1]University of Florida
[2]Kansas State University
[3]University of Massachusetts Lowell
[4]University of New Hampshire

***Abstract*—Threats posed by both power and Electromagnetic (EM) fault injection (FI) are widely investigated in integrated circuit (IC) security. These fault injection attacks exploit the physical behavior of transistors to induce unintended behaviors in circuits. In recent years, researchers have begun to model and simulate both EM and power fault injection and their impacts. However, most research fails to perform cross-domain analysis from the transistor level to the gate level, then to the RT level, and finally to the system level. Therefore, we propose a cross-layer power and EMFI evaluation framework that helps to assess the impact of EM/Power FI from the bottom up. Our framework takes information from all layers in the IC design process into consideration and uses SPICE simulation to emulate the fault induced by EM/Power spikes. We further propose EMFI register bitflip probability equations to quantify, characterize, and verify our results. Finally, we prove the effectiveness of our approach using a range of system-level benchmarks, and our register bitflip probability matches closely with the SPICE simulation result.**

***Index Terms*—Electromagnetic Fault Injection, SPICE simulation, VLSI design, Fault Modeling**

I. Introduction

Electromagnetic Fault Injection (EMFI) is a subset of hardware or physical fault injection techniques, where an electromagnetic pulse is intentionally directed at components of a digital system to induce errors. This approach capitalizes on the vulnerability of electronic circuits to electromagnetic disturbances, a trait that attackers can exploit to circumvent security measures. Over the years, considerable effort has been dedicated to experimenting with physical fault injection on various Application-Specific Integrated Circuits (ASICs) and microprocessors [2], [13], [14]. These experiments often adopt a trial-and-error methodology without systematical modeling. Countermeasures such as EM sensors and detectors are usually implemented in the post-silicon phase [5], which is not ideal as the cost of IC design will grow exponentially in different stages [6], so mitigating EMFI is preferred in the earlier pre-silicon stages. However, modeling and simulating EMFI accurately has always been a significant challenge.

There have been a few works that have successfully modeled and simulated EM fault injection. In the study by Dumont et al. [4], EMFI was categorized into two distinct fault models: the sampling fault model and the timing fault model. The sampling fault model is based on the concept of altering the values sampled by latches. In digital systems, especially in synchronous designs, data is typically sampled at the edge of a clock cycle. The sampling fault model exploits this by using EM/power pulses to cause transient faults at critical moments when data is being sampled. This can lead to the circuit erroneously sampling incorrect values, thus corrupting the data or the state of the system. The timing fault model suggests that undervolting caused by an EM/power pulse will increase the propagation delay of cells, and thus cause timing violations. Both fault models are verified by previous work [3], [4]. However, previous work has been largely proof-of-concept and, therefore, cannot be effectively scaled to practical VLSI designs with many standard cells. To the best of our knowledge, our work is the first to effectively and scalably model the influence of EMFI on registers in a VLSI design.

Our proposed EM Fault Injection assessment framework shown in Figure 1 specifically targets the sampling fault model. With an accurately configured EM fault injection setup, we utilize Simulation Program with Integrated Circuit Emphasis (SPICE) simulations to predict potential bitflip events in targeted registers. In situations where we lack prior knowledge of the fault injection setup, we recommend a simplified workflow for calculating the probability of bitflipping. This workflow includes calculating the probability of a bitflip under a specific EMFI setup and the conditional probability of multiple registers flipping simultaneously.

By integrating all these elements, our framework offers a robust tool for designers, enabling them to quantitatively evaluate and, thereby, enhance the security of their designs against EMFI before they proceed to the silicon fabrication stage. This proactive approach to security is essential in the contemporary landscape of hardware design, where threats are becoming more sophisticated and pervasive. Our main contributions are summarized as follows:

- As the best of our knowledge, we are the first to propose a framework that utilizes SPICE simulation and PDK documents to evaluate if registers in digital designs are susceptible to EM Fault Injection attacks.
- We then propose simplified register bitflip probability equations that are based on the simulation results to quantifiably estimate the likelihood of EMFI occuring.

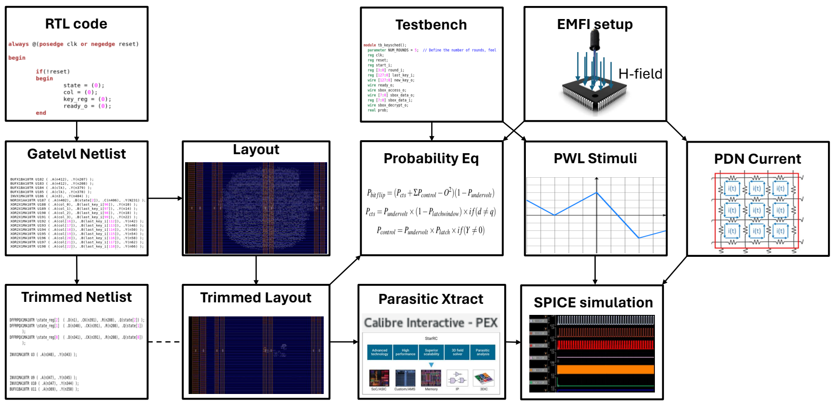


Fig. 1: Overview of our EMFI assessment framework. The framework takes the RTL code, gate-level netlist, the testbench, and the layout from the ordinary digital IC design workflow as input. By extracting the parasitics of the trimmed layout and using the waveforms derived from running the testbench as stimuli, we can accurately model the behavior of standard cells during a EMFI event. Probability equations is also proposed to save the efforts to execute SPICE simulations.

- We verify the proposed simplified workflow and equations by performing 5,000 iterations of SPICE simulation for each benchmark. We find that the mean error between simulation and theoretical equation falls under 10% of the average mean value.

The following sections are organized as follows: in Section II, we introduce the prior work of physical fault injection and guided fault injection research. In Section III, we elaborate on the threat model for this framework. in Section IV, we detail the workflow to model and simulate EM/power FI in a SPICE simulation and propose a simplified workflow of calculating bitflip probability for EMFI events. The results of our framework on 5 different benchmarks are presented in Section V. We further discuss how our proposed probability equation could be used in Bayesian model-based system FI evaluation and hardware fuzzing VI. Section VIII concludes the paper.

II. Related Work

Our work can be used to evaluate the potential of any register to bitflip under a sampling fault. Two previous works [12] and [4] inspire us to develop this workflow.

Dumont et al. first developed a method for simulating and modeling EMFI [4] [3]. These two papers, inspired by previous EM fault model studies [10], [11], categorize the EMFI into sampling fault and timing fault models, and simulate and explain how these faults happen. A tape-out is also done to verify their proposed theory. However, their work does not address the scalability issue. Large designs like most VLSI designs are too large to be simulated by the method they propose. Utilizing commercial VLSI EDA tools for scalable physical fault injection is critical.

Pundir et al. introduced a workflow to simulate faults caused by laser FI in [12]. They validate their framework by testing on an AES design and successfully cause Differential Fault Attacks. This work can handle large VLSI design, which in some way addresses the scalability issue. Our work, on the other hand, is trying to build a similar but more accurate workflow featuring SPICE simulation to simulate fault caused by EMFI.

In [17], a scalable system-level methodology is introduced for estimating system reliability through low-level hardware fault injection. This assessment technique offers precise calculations by leveraging the masking probability inherent between hardware and software constituents. However, confining the analysis solely to low-level technological effects proves insufficient for a thorough exploration the impact of FI.

In the realm of identifying and mitigating fault attacks, [9] established a security-aware Finite State Machine (FSM) encoding system that incorporates both protected states and transitions. This proposed FSM architecture restricts access to protected states exclusively for authorized states, forbidding access from both unauthorized states and don't-care states. Furthermore, the model reduces the incidence of don't-care states, thereby creating an opportunity for the integration of an additional FSM encoding system to augment system stability.

In addition, there are existing works that focus on utilizing physical information to develop fault injection attacks on embedded devices. For example, Jain *et al.* [7] introduced a novel differential fault analysis attack methodology that uses stuck-at fault patterns to determine the secret key of a locked circuit. Luo *et al.* [8] proposed a remotely-guided fault injection attack, known as Deepstrike, which applies power glitching fault injections to disrupt the functionalities of DNN accelerators deployed on the Cloud-FPGA platform. Most recently, Zhong *et al.* [19] utilized a specific input pattern that sensitizes a key bit to the primary output. Consequently, their resulting fault attack is capable of breaking any locking technique that relies on a stored secret key.

III. Threat Model

In our research, we focus on developing a defense mechanism against EMFI attacks. This effort culminates in a post-layout, pre-silicon security-driven evaluation framework that is specifically tailored for these kinds of threats and can be integrated in the existing EDA toolkits. Our work is primarily intended to assist designers in assessing the vulnerability of their digital designs to EM and power fault injection attacks, which are increasingly prevalent in the field of hardware security. We assume that the designer has detailed knowledge of the layout information, which provides insight into the physical arrangement and connections of the circuit components. In addition, a thorough familiarity with the testbench information is required. The testbench information is crucial as it outlines the environment or conditions under which the design is tested and evaluated, thereby determining its resilience to the aforementioned attacks. Lastly, gate-level netlist information is also a vital component of our framework. The gate-level netlist presents a detailed description of the electronic circuitry at the logical gate level, offering an in-depth view of the circuit's functional and timing aspects.

By integrating all these elements, our framework can evaluate the susceptibility of registers bitflipping caused by EMFI in a VLSI design.

IV. Methodology

In what follows, we introduce our workflow of using SPICE simulations to model IC behavior under power and EMFI attacks. We will introduce each step in sequence in this section. We also include a demo analysis of the keyschedule module in the SystemC-AES benchmark to better explain each step in our EMFI assessment framework. The state[2:0] FSM in the keyschedule model only takes a clock signal and reset signal as input, and will repeat the state transition 0-1-2-3-4-0 loop after the circuit is powered up. Any state larger than 4 is an undefined state and Don't Care Transitions (DCTs) may occur once the FSM enters an undefined state, as shown in Figure 2. Undefined states may serve as a hidden triggers for Hardware Trojans that can never be reached through normal functional test, such that DCTs can be exploited to bypass normal authentication processes and directly enter sensitive states that may cause confidential information leakage [1].

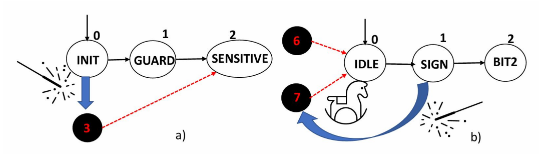


Fig. 2: Don't Care Transition in vulnerable designs.

A. Layout and Netlist Trimming

The refined layout and gate-level netlist must encompass all fan-in cells associated with the targeted registers, as shown in Figure 3. This approach aligns with the methodology used in [9], which involves retaining all fan-in key bits linked to an output. For instance, consider the SystemC-AES benchmark. Here, we focus on a FSM with a 3-bit state, designating these three registers as our targets. The fan-ins for this setup comprise 13 combinational logic gates and 3 registers, along with 5 Clock Tree Synthesis (CTS) buffers. This process of trimming can be efficiently executed using widely-used commercial EDA tools, such as Synopsys Design Compiler.

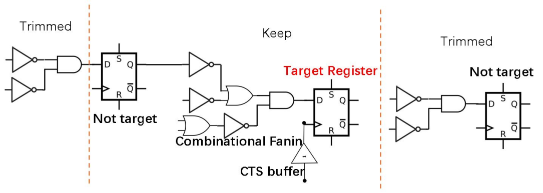


Fig. 3: Gate-level schematic demonstrating the trimming strategy.

Continuing with netlist trimming, we employ a consistent strategy for the layout phase. This assumes that designers have access to both the layout in GDS binary file format and the design files used in automatic Place and Route software. In practice, this involves deleting standard cells from the layout that were eliminated during the netlist trimming phase. Concurrently, we also remove all wiring connections related to these trimmed cells. During post-trimming, the layout is streamlined to include only the Power Distribution Network (PDN) structure and those cells logically connected to the target registers, ensuring a focused and efficient design. Figures 4 and 5 show the layout before and after the trimming process. The trimming process is to lower the computational load for later parasitic extraction and SPICE simulation.

B. PDN Parasitics Extraction and EMFI Modeling

Upon completing the layout trimming process, the next step in our methodology is to import the trimmed layout into an analog layout design environment. This environment is typically integrated with parasitic extraction tools, such as Calibre PEX or StarRC, which are instrumental in accurately modeling the parasitic effects present in the layout. To facilitate straightforward voltage measurements, output ports are added to the power supply of each cell in the layout. Subsequently, the input power IO pads are designated as input nodes for both VDD (positive supply voltage) and VSS (ground), ensuring a comprehensive setup for accurate power delivery and measurement. Once these preparatory steps are completed, we engage the parasitic extraction tools to extract the resistance (R) and capacitance (C) values of the layout. This information is crucial in generating new SPICE models, which will serve as inputs for subsequent stages of our process. This approach allows us to model our designs with enhanced accuracy, taking into account the real-world physical and electrical characteristics that would impact the performance in an actual silicon implementation.

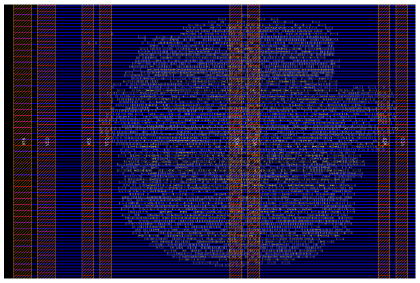


Fig. 4: SystemC AES benchmark layout before trimming (without routing wires).

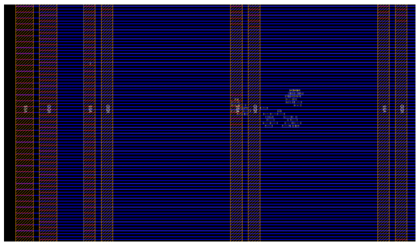


Fig. 5: SystemC AES benchmark layout after trimming (without routing wires).

C. SPICE Simulation

After layout trimming and parasitic extraction of the layout, we run the testbench of the designs to generate `vcd` waveform files. Then we use an open source tool to transform the `vcd` files into piecewise linear (PWL) files for each standard cell to setup the SPICE simulation stimuli [15]. The toggling activities of the trimmed cells could also cause voltage fluctuations that influence the behaviors of untrimmed cells. This is also modeled by treating every trimmed cell as a current source with the value being a pulse during these switches. The pulse magnitude and frequency is derived from simulations of each kind of standard cells switching between logic 0 and 1s.

We also need to generate stimuli on PDN based on different FI setups before simulation. For power fault injection, we simply change the voltage waveform at the input nodes and through the RC network of the PDN, the supply voltage fluctuation of every cell in the trimmed netlist can then be simulated during the SPICE simulation. For EM fault modeling, we add a current source for each extracted resistor of the PDN segments that are affected. The current injected is equal to the induced current by the EMFI. This current magnitude is a determined fixed value in this work, but can be changed by using output from EM simulating tools like Ansys HFSS. Finally, we run the SPICE simulation and check the waveform of the target register to see if any fault occurs.

D. Register Bitflip Probability

Before introducing the likelihood of targeting FI in a single register, we present two observations upon which we base our analysis. Based on these observations, we propose bitflipping probability equations to simplify the EMFI assessment framework. We favor these equations over SPICE simulations due to the extensive time required for running numerous simulations; it typically takes about one day to verify one benchmark on our server equipped with 8 Intel i7 cores using Cadence Virtuoso. Moreover, since IC designers may like to check design robustness against EMFI before steps like parasitic extraction and verification which are more costly in the IC design lifecycle, we advocate for a simplified workflow that incorporates the layout, netlist, PDN topology, and the EMFI setup. The probability is determined by an unguided EMFI event. We, therefore, propose a probability equation to calculate the likelihood of a single register flip and a conditional probability equation to assess the likelihood of multiple simultaneous bit-flipping events.

Observation 1: When undervolting occurs on all cells. This case is likely to happen when power FI happens. The pulse injected at power pins will propagate through the whole PDN network and almost every cell that is logically connected to the target register will be influenced. If the latency caused by the RC of the PDN network is small, then most cells will be undervolted at the same time. Under this condition, the target register will most likely not be flipped because when the fault value arrives at the target register, the register itself is also being undervolted and thus not functioning. Therefore, there is a probability of a bitflipping event on target register only if the RC network causes large delay. In Figure 6, V_{fault} is applied on the source of power supply VSS and influences all cells with a small RC delay, so that all three registers of state[2:0] are undervolted and stop functioning meaning a bitflip never occurs.

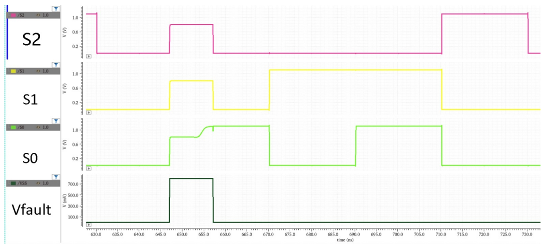


Fig. 6: Observation 1: when power FI happens, the fault event is likely to influence most cells and if the delay caused by RC network is small, it is hard to cause a bitflip.

Observation 2: When undervolting occurs on several control signal cells. When undervolt events only affect several PDN segments, usually during an EMFI attack where the probe is near the IC surface, the RC network of the PDN has less impact on inducing a fault. Rather, the physical location of the injected currents will decide if the target register will flip. The target register itself will likely be functioning normally and sampling voltages. Undervolting on both control signal gates and clock buffers will cause the target register to flip if the undervolt propagates to the register inside the latch window. In Figure 7, V_{fault} only influences the control signal cells of state[2] and thus only state[2] is flipped while the other two registers function properly, making the FSM enter the undefined state 5. Control signal gates are defined by whether the gate is toggled, meaning the value change could propagate to the target register without being stopped at some logic gates in the middle. Specifically, if X is an inverter, then the fan-in signal is the control signal; if X is a OR(NOR included) gate, then in its multiple fan-ins; if all fan-ins except $Input_i$ are logic 0, then $Input_i$ is the control signal. If X is an AND (NAND included) gate, if all fan-ins except $Input_i$ are logic 1s, then $Input_i$ is the control signal. If X is an XOR gate, then all fan-ins are control signals. Moreover, control signal of the control signal gates of X is the control signal of X. Figure 8 is the showcase of how we extract control signals in red by looking the internal states. Any flip in the red signals will result in a change in the D port of the target register, while blue signals will not.

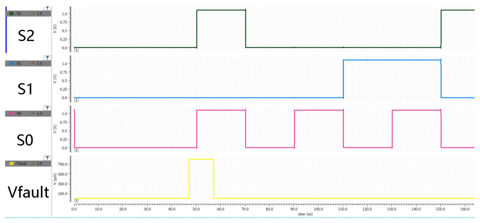


Fig. 7: Observation 2: when EMFI happens, the fault event is likely to influence fewer cells and easier to cause a bitflip.

1) Probability for inducing FI in a single register: It's important to note that in the case of Power FI, the latency introduced by the RC network across the entire PDN must be considered, which complicates simplification. Therefore, the equations provided below in section IV-D1 and section IV-D2 are applicable only to EMFI scenarios.

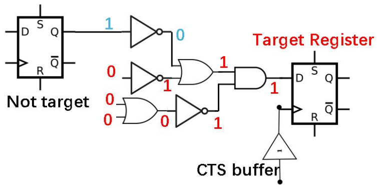


Fig. 8: Determining which cells are control signal gates based on current internal states, control signals are labelled red and non-control signals are labelled blue

From the simulation results we observe in Observation 1 and 2 in Section IV-D, we can conclude that for a sampling fault to happen, it must meet two conditions. First, the targeted register itself should not be undervolted and, secondly, either the control signal or clock buffer should be undervolted. Based on these, we propose equation 1, 2, and 3 to simplify the framework, saving the effort from running computationally heavy SPICE simulations.

$$P_{bitflip} = (P_{cts} + \Sigma P_{control} - O^2)(1 - P_{undervolt}) \tag{1}$$

$$P_{cts} = P_{undervolt} \times (1 - P_{latchwindow}) \times if(d \neq q) \tag{2}$$

$$P_{control} = P_{undervolt} \times P_{latch} \times if(Y \neq 0) \tag{3}$$

In Equation 1, P_{cts} denotes that the bitflip is caused by an additional fault clock due to undervolting on the CTS buffer. $P_{control}$ denotes that bitflipping is caused by undervolting on one of the control signal gates. O^2 denotes the condition that two or more control signal gates are undervolted at the same time, but we assume the probability of this kind is small because it requires two undervolt events to occur at the same time and can be omitted for simplicity for later calculation. $1 - P_{undervolt}$ denotes the probability of the target register not being undervolted.

Equation 2 shows the probability of a bitflip caused by the CTS buffer being undervolted. $P_{undervolt}$ denotes that the PDN segment of the CTS buffer must be undervolted. $1 - P_{latchwindow}$ denotes that this undervolt event must happen outside the latch window, i.e. rising edge of the normal clock cycle. If it happens within the latch time window, then the faulted pulse will just cover the original rising clock edge rather than form a new additional rising clock edge. $if(d \neq q)$ denotes that when the additional rising clock edge arrives at the target register, the sampling value d, or input, of the register must be different from the latched value q, or output. Otherwise, the latched value q will not flip but keep its original value.

Equation 3 shows the probability of a bitflip caused by control signal gates being undervolted. Again, $P_{undervolt}$ denotes that the PDN segment of one control signal gate must be undervolted. P_{latch} denotes that when the undervolt event happens and the faulted value propogates to the targeted register, this register must happen to be inside the latch window, otherwise the faulted value would not be sampled. $if(Y \neq 0)$ denotes that when the control signal gate is undervolted, the original value of the control signal gate should not be logic 0. In other words, the undervolt event can generate a false logic 0 only when it is not logic 0.

2) Conditional Probability considering multiple bitflip events: It is far from enough to only know the probability of inducing a bitflip on a single register when evaluating the EMFI susceptibility of a design. Usually, a secure state comprises multiple secure registers. To expand the ability to assess the design on a higher level, we need to introduce conditional probability considering multiple bitflip events in our framework.

To showcase how to calculate conditional probability, we here assume two target registers X and Y. Fan-in cells to register X are the CTS buffer CTS_X and control signal gates X_i. Fan-in cells to register Y are the CTS buffer CTS_Y and control signal gates Y_i. Based on what Section IV-D1 introduces, the probability equations for register X and Y are shown in Equation 4 and 5, respectively.

$$P(X) = (P_{cts}(CTS_X) + \Sigma P_{control}(X_i))(1 - P_{undervolt}) \quad (4)$$

$$P(Y) = (P_{cts}(CTS_Y) + \Sigma P_{control}(Y_i))(1 - P_{undervolt}) \quad (5)$$

There are two conditions noteworthy: 1) that they do not share any CTS buffer or control signal gates in common or in the same PDN segment, and 2) that gates are shared or in the same PDN segment. If they do not share any CTS buffer or control signal gate in the same PDN segment, then:

$$P(XY) = P(X)P(Y) \quad (6)$$

$$P(Y|X) = P(Y) \quad (7)$$

If they share some control signal gates, or their control signal gates happen to be placed in the same PDN segment, then the equation is different. Let us assume control signal gates X_1 and Y_1 are the shared gates, then the equation changes to:

$$P(XY) = (P(X) - P(X_1))(P(Y) - P(Y_1)) + P(X_1) \quad (8)$$

where

$$P(X_1) = P_{undervolt} \times P_{latch} \times if(X_1 \neq 0) \quad (9)$$

and

$$P(Y|X) = \frac{(P(X) - P(X_1))(P(Y) - P(Y_1))}{P(X)} + \frac{P(X_1)}{P(X)} \quad (10)$$

Equation 6 and 7 describe the probability of independent events happening at the same time. If two registers have no control logic in common, or have no control logic that resides in the same PDN segment, then the single register bitflip probabilities of these two or more registers are independent. Equation 8, 9, and 10 describe the probability of dependent single register bitflip events. They share common parts of control logic and, thus, follow the conditional probability equation that takes the shared factors into consideration [18].

V. Experimental Results

We evaluate this framework on 5 benchmarks covering DCTs and undefined states. For each benchmark, the vulnerable undefined state can be reached by EM fault injection. We calculate the bitflip probability based on the equation we derive from SPICE simulations to evaluate if the equations derived in Section IV-D are accurate.

A. Benchmark Selection and Targeted FSM

We evaluate the proposed framework across 5 distinct benchmarks, each carefully chosen to explore influence of EMFI on the handling of DCTs and undefined states in FSMs. These benchmarks represent a diverse array of systems and applications, offering a comprehensive assessment of our framework's versatility and robustness. These benchmarks are examined in [1] to determine wheter DCT vulnerabilities are able to access secure states. They include:

1) APB2SPI - This benchmark targets a state variable named 'STATE'. It exhibits 3 DCTs at both the Register-Transfer Level (RT-Level) and the Gate-Level, transitioning from state [3] to [0,2].
2) RS232 UART - In this benchmark, the 'transmit' module houses the DCT FSM with a state variable named 'state'. It provides insights into serial communication protocols.
3) UART - With 'recv-state' as its state variable, this benchmark delves into aspects of Universal Asynchronous Receiver-Transmitter (UART) systems.
4) IMA-ADPCM-ENC - Highlighting audio encoding, this benchmark's state variable is 'pcmSq'. It displays 2 DCTs at the RT-Level and 12 at the Gate-Level, transitioning from states [6,7] to [0] and [0,5], respectively.
5) SystemC AES - With state in submodule 'keyschedule' containing a DCT FSM, the state variable is 'state' in the module. This FSM have DCTs from state[5] to [0,4] as inspected by Ruochen et al. [1]

Each benchmark encompasses netlists at both the RT-Level and the Gate-Level, derived from YOSYS, a synthesis framework. This selection of benchmarks, encompassing diverse aspects like system control, encryption, and communication protocols, provides a rigorous testing ground for evaluating our framework's efficacy in handling DCTs and undefined states in FSMs. For our simulation work, we implement these benchmarks using TSMC 65nm PDK.

B. SPICE Simulation and Probability Result

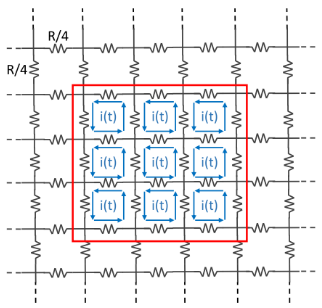
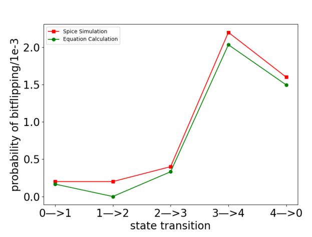


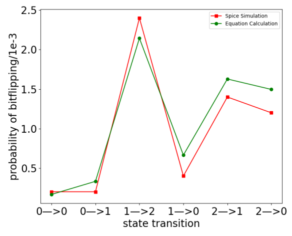
Fig. 9: PDN segments injected current when Probe close to IC surface [4]

For SPICE simulation, we export the simulated voltage waveform as CSV files and count the bitflip events using Python scripts. These statistics are sorted by different FSM state transitions. For the probability calculation, we hard-code the equation in verilog in the testbench of these designs. By comparing the probability and how many bitflip events happen in the simulation, we are able to evaluate the effectiveness of the proposed equation to simplify EMFI vulnerability assessment.

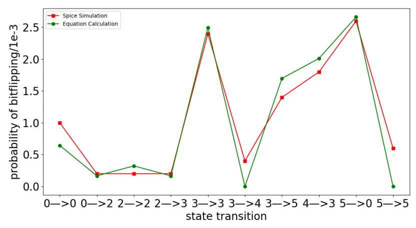
For both SPICE simulation and probability calculation, we assume similar EMFI modeling as used in the paper by



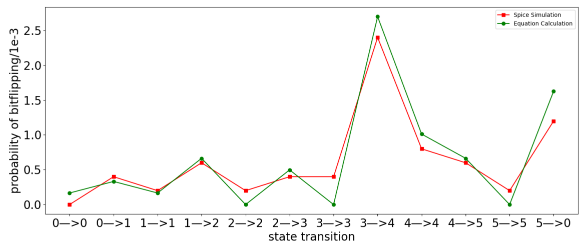
(a) Register state[2] of FSM in SystemC AES



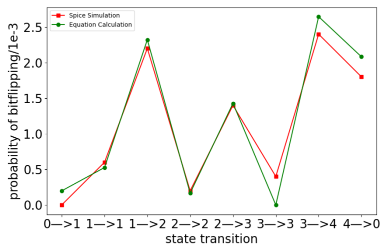
(b) Register STATE[1] of FSM in APB2SPI



(c) Register state[2] of FSM in RS232 U-XMIT verilog



(d) Register pcmSq[2] of FSM in IMA ADPCM-ENC



(e) Register recvstate[2] of FSM in UART-1

Fig. 10: Bitflipping probability calculation results vs. SPICE simulation results for 5 benchmarks, mean error falls under 10% of the mean value

Dumont et al. [4] in which the probe is very close to the IC surface and only a few PDN segments are undervolted as shown in Figure 9. Our benchmark layouts all have 6000 segments in total and we assume only 10 are influenced to a level that undervolting can be wrongly sampled. $P_{undervolt}$ is 1/600 then. The clock frequency is 100MHz, and the simulated FI pulse width is 1ns, which makes $P_{latchwindow}$ 1/10. The results are shown in Figures 10a, 10c, 10e, 10d, and 10b. The mean error falls under 10% of the mean value for each benchmark.

After we verify the effectiveness of the proposed bitflip probability calculation equation for a single register, we then conducted additional simulations in Cadence Virtuoso to verify our conditional probability equation for multiple registers. We target the state in keyexpand submodule in the SystemC-AES benchmark and try to let the FSM directly enter state 5 from other states. To achieve this, multiple states may need to be flipped at the same time. For example, if we want to enter undefined state 5 from state 0, we will need to flip both state[2] and state[0] at the same time during the normal transition from 0 to 1. From the results shown in Figure 10a, we already know the bitflip probability of state[2] in all 5 legal transitions. We, therefore, can start from this state and get the probability of transitioning to state 5 during these 5 legal transitions. To verify the conditional probability, we perform 50,000 rounds of SPICE simulations to achieve a higher resolution to capture two or more registers flipping at the same time. This number is chosen considering the trade-off between resolution and computation time of SPICE simulation. The result of this conditional probability versus simulation is shown in Figure 11. The most vulnerable transition is during the legal state transition from 3(011) to 4(100). To enter undefined state 5(101), two registers state[2] and state[1] need to flip simultaneously, which is the easiest to trigger according to both simulation and our analysis. The mean error of calculating two and more registers flipping at the same time, which falls under 30% of the mean value, is larger than single registers because the error would aggregate during the multiplication of the probabilities of single register bitflips.

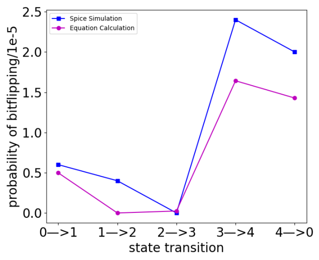


Fig. 11: The probability of the FSM entering undefined state 5 from both simulation and calculation, system C AES benchmark

C. Discussion on the mismatch between Equation and Simulation

Two error sources cause the mismatch between the equation and the simulation. One source is the O^2 probability in Equation 1 that denotes the condition that two or more control signal gates in different PDN segments are undervolted at the same time. Our equation omits this situation and adds up the probability of each the control signal gates being undervolted individually, so the result from the equation would be slightly higher as the equation may take the O^2 condition into consideration multiple times.

The other source is to assume the probability of each PDN segment being undervolted is identical. In SPICE simulation, the PWL file-driven current sources help simulate the voltage fluctuation caused by currents from other trimmed cells, but in our analysis, the voltage fluctuation is omitted. Voltage fluctuation will cause the cells to be flipped more easily because the dropping VDD is closer to the threshold of having an error and thus the circuit would be more sensitive to fault injection attacks. Generally speaking, this source of error causes the equation result to be lower than actual simulations.

VI. Discussion

Component-based Bayesian Model for Cross-layer Evaluation. Vallero et al. [17] propose a scalable system-level methodology designed to estimate system reliability by conducting low-level hardware fault injection. This assessment approach provides accurate estimations by utilizing the inherent masking probability that exists between hardware and software components. Our work can be combined with this work to extend the bottom layer from gate-level in [17] to SPICE/transistor-level in this work.

Hardware Fuzzing. Hardware Fuzzing, proposed by Trippel et al. [16], translates RTL hardware to a software model and fuzzes the software directly. Hardware Fuzzing can reach higher coverage of FSM states within less time, thus can be used to replace and improve the golden standard–dynamic hardware testing. The probability of register bitflips can be integrated by specifying the probability of each register being flipped. In this way, the original hardware fuzzing work can consider physical FI and provide new improved test patterns. We will explore the combination of the EMFI assessment framework and Hardware fuzzing as our future work.

VII. Acknowledgement

This research was partially supported by National Science Foundation under Award Number 2019283 and partially supported by Office of Naval Research under award number N000142412048.

VIII. Conclusion

In this paper, we proposed a framework to evaluate the susceptibility of a digital design to physical fault injection attacks, we further propose a simplified bitflipping probability calculating method as a metric for such evaluation. The proposed framework is the first in the field, to our knowledge, to provide a method that analyzes information cross-domain, from layout to RTL level, to assess the security level of scalable digital designs. The framework is evaluated on 5 different benchmarks and the average difference between probability calculation and SPICE simulation falls under 10%. Furthermore, we discuss how a probability can be used in a larger physical FI assessment framework that finally evaluates on the software level by using the Bayesian model or by combining hardware fuzzing technique to discover vulnerabilities at a higher level.

References

[1] Ruochen Dai and Tuba Yavuz. A symbolic approach to detecting hardware trojans triggered by don't care transitions. ACM Transactions on Design Automation of Electronic Systems, 28(2):1–31, 2022.

[2] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, Philippe Orsatelli, Philippe Maurine, and Assia Tria. Injection of transient faults using electromagnetic pulses practical results on a cryptographic system. ACR Cryptology ePrint Archive (2012), 2012.

[3] Mathieu Dumont, Mathieu Lisart, and Philippe Maurine. Electromagnetic fault injection: How faults occur. In 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pages 9–16. IEEE, 2019.

[4] Mathieu Dumont, Mathieu Lisart, and Philippe Maurine. Modeling and simulating electromagnetic fault injection. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 40(4):680–693, 2020.

[5] David El-Baze, Jean-Baptiste Rigaud, and Philippe Maurine. An embedded digital sensor against em and bb fault injection. In 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pages 78–86. IEEE, 2016.

[6] Xiaolong Guo, Raj Gautam Dutta, Yier Jin, Farimah Farahmandi, and Prabhat Mishra. Pre-silicon security verification and validation: A formal perspective. In Proceedings of the 52nd annual design automation conference, pages 1–6, 2015.

[7] Ayush Jain, M Tanjidur Rahman, and Ujjwal Guin. Atpg-guided fault injection attacks on logic locking. In 2020 IEEE Physical Assurance and Inspection of Electronics (PAINE), pages 1–6. IEEE, 2020.

[8] Yukui Luo, Cheng Gongye, Yunsi Fei, and Xiaolin Xu. Deepstrike: Remotely-guided fault injection attacks on dnn accelerator in cloud-fpga. In 2021 58th ACM/IEEE Design Automation Conference (DAC), pages 295–300. IEEE, 2021.

[9] Adib Nahiyan, Farimah Farahmandi, Prabhat Mishra, Domenic Forte, and Mark Tehranipoor. Security-aware fsm design flow for identifying and mitigating vulnerabilities to fault attacks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 38(6):1003–1016, 2019.

[10] Sébastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine. Em injection: Fault model and locality. In 2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pages 3–13. IEEE, 2015.

[11] Sébastien Ordas, Ludovic Guillaume-Sage, Karim Tobich, J-M Dutertre, and Philippe Maurine. Evidence of a larger em-induced fault model. In Smart Card Research and Advanced Applications: 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers 13, pages 245–259. Springer, 2015.

[12] Nitin Pundir, Henian Li, Lang Lin, Norman Chang, Farimah Farahmandi, and Mark Tehranipoor. Security properties driven pre-silicon laser fault injection assessment. In 2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pages 9–12. IEEE, 2022.

[13] Lionel Riviere, Zakaria Najm, Pablo Rauzy, Jean-Luc Danger, Julien Bringer, and Laurent Sauvage. High precision fault injections on the instruction cache of armv7-m architectures. In 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pages 62–67. IEEE, 2015.

[14] Jörn-Marc Schmidt and Michael Hutter. Optical and em fault-attacks on crt-based rsa: Concrete results. na, 2007.

[15] SteveMSong. Vcd file to pwl file script. https://github.com/SteveMSong/python-scripts/blob/master/vcd-pwl.py.

[16] Timothy Trippel, Kang G Shin, Alex Chernyakhovsky, Garret Kelly, Dominic Rizzo, and Matthew Hicks. Fuzzing hardware like software. In 31st USENIX Security Symposium (USENIX Security 22), pages 3237–3254, 2022.

[17] A. Vallero, A. Savino, G. Politano, S. Di Carlo, A. Chatzidimitriou, S. Tselonis, M. Kaliorakis, D. Gizopoulos, M. Riera, R. Canal, A. Gonzalez, M. Kooli, A. Bosio, and G. Di Natale. Cross-layer system reliability assessment framework for hardware faults. In 2016 IEEE International Test Conference (ITC), pages 1–10, 2016.

[18] Wikipedia. Conditional probability, 2024.

[19] Yadi Zhong, Ayush Jain, M Tanjidur Rahman, Navid Asadizanjani, Jiafeng Xie, and Ujjwal Guin. Afia: Atpg-guided fault injection attack on secure logic locking. Journal of Electronic Testing, 38(5):527–546, 2022.