Characterizing and Modeling Synchronous Clock-Glitch Fault Injection

Amélie Marotta[1], Ronan Lashermes[1], Guillaume Bouffard[2], Olivier Sentieys[1], and Rachid Dafali[3]

[1] University of Rennes, Inria, Rennes, France
amelie.marotta@inria.fr, ronan.lashermes@inria.fr, olivier.sentieys@inria.fr
[2] National Cybersecurity Agency of France (ANSSI), Paris, France
guillaume.bouffard@ssi.gouv.fr
[3] DGA-MI, France
rachid.dafali@def.gouv.fr

Abstract. In the realm of fault injection (FI), electromagnetic fault injection (EMFI) attacks have garnered significant attention, particularly for their effectiveness against embedded systems with minimal setup. These attacks exploit vulnerabilities with ease, underscoring the importance of comprehensively understanding EMFI. Recent studies have highlighted the impact of EMFI on phase-locked loops (PLLs), uncovering specific clock glitches that induce faults. However, these studies lack a detailed explanation of how these glitches translate into a specific fault model. Addressing this gap, our research investigates the physical fault model of synchronous clock glitches (SCGs), a clock glitch injection mechanism likely to arise from EMFI interactions within the clock network. Through an integrated approach combining experimental and simulation techniques, we critically analyze the adequacy of existing fault models, such as the Timing Fault Model and the Sampling Fault Model, in explaining SCGs. Our findings reveal specific failure modes in D flip-flops (DFFs), contributing to a deeper understanding of EMFI effects and aiding in the development of more robust defensive strategies against such attacks.

Keywords: Fault Injection, Clock Glitch, Physical Fault Model

1 Introduction

The majority of the electronic products used in our daily lives manipulate, store, and transmit sensitive data. The largest part of these products are designed without taking into account the threats generated by fault injection (FI) attacks. To define countermeasures against FI, a full characterization of the fault effects must firstly be conducted. This characterization remains a complex and challenging task because it must be analyzed at various levels (model, microarchitecture, gate and transistor).

Various means are used to conduct FI attacks, such as lasers [12] and electromagnetic [14] injections. However, electromagnetic fault injection (EMFI) [20] is particularly noteworthy as a frequently employed mean due to its minimal setup requirements for targeting a component. In this article, our primary focus is on studying the effects of EMFI attacks. However, EMFI attacks are characterized by their lack of precision, resulting in a broad and difficult-to-control impact, potentially affecting a wide range of elements embedded in the targeted component.

Previous studies [5, 25] have shed light on the EMFI impact on phase-locked loops (PLLs). They have brought to attention the generation of clock glitches induced by EMFI, which lead to faults in components. While this research has been pivotal in understanding the effects of EMFI, it falls short of an in-depth explanation of how these clock glitches concretely translate into faults.

Several articles [7, 8, 9, 13, 16] have proposed low-level fault models to explain the occurrence of faults induced by EMFI. However, these models focus on the specific case where EMFI interact with power and ground signals.

Consequently, there remains a gap [15] in our comprehension of why EM-induced clock glitches result in specific fault perturbations as it is difficult to isolate them from the predominant effects (power and ground interaction).

Within this context, this article aims at providing a low-level fault model that explain how EM-induced clock glitches lead to faults. However, we use a clock glitching platform and not EMFI to induce faults, allowing us to have only faults of interest. Through physical experimentations and simulations, we highlight the main mechanism of fault as well as influencing factors.

This article is organized as follows. Section 2 reviews the related works concerning EM injection and clock glitch attacks and the associated fault models. We conclude that no low level fault model in the literature can be applied to our case. Therefore we state the hypotheses behind our own model dubbed controlled synchronous clock glitch in Section 3. To verify these hypotheses, we conduct experiments and simulations in Section 5, with a setup described in Section 4. These experiments allow to develop an understanding of the different factor that may lead to a fault.

2 Related works

2.1 Overview of Fault Injection Analysis

By utilizing different FI methods and parameters, attackers can achieve various effects. Characterizing these effects aims to develop a fault model that represents what happen at specific abstraction levels. Therefore, understanding the effects of faults is crucial for implementing efficient countermeasures. The fault model is usually characterized at the physical, register-transfer, or microarchitectural levels, as described in [26]. First, the physical fault model analyzes the interaction between FI and transistors and logic gates. At this level, the goal is to understand why the photons injected from a laser pulse can switch a logic gate output, or why a D flip-flop (DFF) samples an incorrect value under EMFI. This level considers the analog nature of electrical current and voltage signals [8, 9, 16]. Second, at the register-transfer level (RTL), a fault is modeled as a logic signal alteration. Here, the analysis focuses on how a bit flip or a 'stuck at 0' (or 1) propagates through a circuit. Finally, at the microarchitectural level, a fault is analyzed by its impact on the microarchitecture. For instance, a bit-flip on the forwarding control signal can lead to an instruction skip [21]. Microarchitectural fault models include instruction-set architecture (ISA) fault models that represent a fault as an instruction modification. In other words, at the ISA level, the consequence of a fault can be linked to one instruction being transformed into another [22, 23]. Some microarchitectural faults cannot be modeled at the ISA level. For example, in [24], faults impact the data cache, but the instructions remain intact.

This paper focuses on analysing the physical fault model arising from a specific EM-induced clock glitch, described in Section 2.2. The study intentionally excludes RTL and microarchitectural fault models from its scope.

2.2 EMFI on the PLL

To the best of our knowledge, the first mention of the influence of EMFI on the PLL was in [25]. The PLL is a component that takes a low-frequency clock signal as input and generates a high-frequency, stabilized clock signal. The objective was to use the PLL as a detector for EMFI. The authors consider the booting-up phase of the PLL, where it transitions from an "*unlocked*" state to a "*locked*" one. This transition does not take the same amount of clock cycles with and without EMFI, thus demonstrating the sensitivity of the PLL to EMFI.

Claudepierre *et al.* explain in [5] that EMFI alters the behavior of the PLL and generates clock glitches. More precisely, the injection modifies a clock cycle: the rising edge does not reach the high state because the injection causes a drop in the signal until the next clock cycle. The glitched clock cycle delivers less energy (it has a lower voltage for a shorter duration), but remains synchronous. The authors also show that the injection may eliminate the cycle altogether, suggesting that the characteristics of glitched clock cycles may vary between injections. In this paper, this specific clock glitch is referred to as synchronous clock glitch.

2.3 TRAITOR

To reproduce EMFI clock cycle perturbations, Claudepierre *et al.* introduced an FI tool named TRAITOR [6]. TRAITOR can control the amplitude parameter, which defines the energy level of the synchronous clock glitch. This allows for more precise control over the glitch. For the remainder of this article, this perturbation is referred to as controlled synchronous clock glitch (CSCG). Since there is currently no method to demonstrate equivalence, we refer to EM-induced clock glitches as synchronous clock glitch and TRAITOR-induced clock glitches as CSCG.

In their study, Claudepierre *et al.* targeted a microcontroller to analyze the TRAITOR fault model [6]. The primary induced microarchitectural fault model is an instruction skip. As a result, with its very high success rate and ability to perform a large number of faults, TRAITOR is a suitable FI tool for `NOP`-oriented programming, as explained in [17]. By carefully replacing selected instructions with a `NOP` assembly instruction, an attacker can modify a running program, akin to a Returned-Oriented Programming (ROP) attack. As demonstrated by Gicquel *et al.* in [10], without appropriate hardware countermeasures, such an attack is almost guaranteed to succeed.

TRAITOR can be used to simulate the effect of EMFI. Several fault models have been proposed to explain the impact of EMFI on circuits, and these may apply to TRAITOR. However, the physical model of such EM-induced fault has not yet been analyzed, to our best knowledge.

2.4 Known Fault Models

In this paper, an error is defined as an incorrect transient value, for example, when an induced current in a wire modifies its logic state. A fault occurs when an error propagates up to a memory element, allowing the propagation of the incorrect value into the next clock cycle.

For a fault to have a lasting impact on a circuit, an erroneous value must be stored at some specific point. For instance, a current induced by laser FI may create a latch-up, which forms the basis of the stuck-at fault model [18]. In most cases, the storage of an error results from the interaction of transient signals with a positive-edge-triggered DFF. This section discusses the correct behavior of the DFF and known failure modes as described in the literature.

The DFF's Correct Behavior Consider a simple circuit composed of two DFFs with some logic in between, as depicted in Figure 1. For proper sampling (or storage) to occur from $\mathtt{D_1}$ to $\mathtt{Q_1}$ in the second DFF, the data coming from $\mathtt{D_1}$ must be stable during the setup and hold time window, defined by $\mathtt{t_{setup}}$ before and $\mathtt{t_{hold}}$ after the rising edge of `clk`, respectively. For the fault models presented in the following sections, this DFF is assumed to be under fault.

Timing Fault Model The Timing Fault Model was the first of its kind to be proposed. In 2008, Selman *et al.* [19] demonstrated that underpowering a circuit could lead to errors due to setup time violations. Subsequently, in 2010, Agoyan *et al.* [3] showed that shifting a clock's rising edge in time can trigger similar effects. As illustrated in Figure 2, $\mathtt{D_1}$ is unstable during the $\mathtt{t_{setup}}$ time. This violation of the timing constraint can cause $\mathtt{Q_1}$ to enter a metastable state, potentially leading to a fault. In other words, when the input timing constraints are not met, the value of $\mathtt{Q_1}$ becomes non-deterministic. Metastability refers

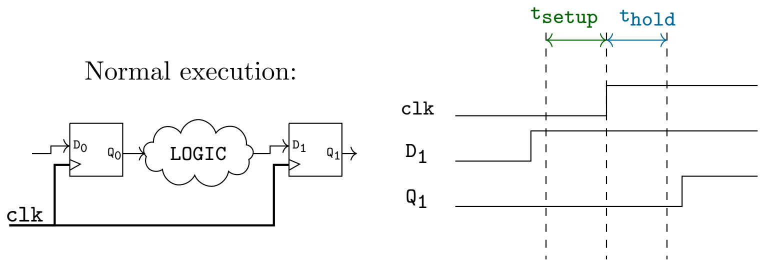


Fig. 1: Normal execution of a simple synchronous circuit.

to the phenomenon where a non-deterministic output is generated if the DFF signal constraints are not adhered to. This setup time violation can occur due to a reduction in the supply voltage of the logic, which extends its execution time (for instance, by underpowering the circuit), or by advancing a clock cycle, resulting in the logic having insufficient time to execute before the next rising edge. This model was initially suggested [7] to explain the effects of EMFI.

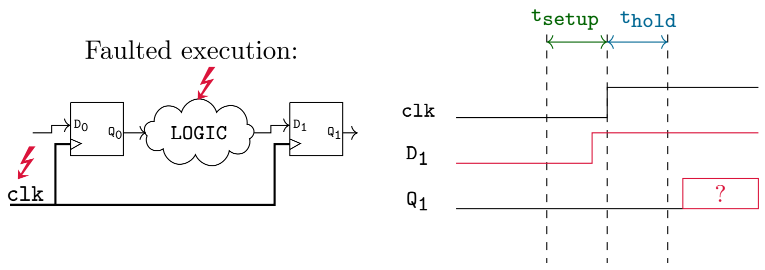


Fig. 2: Timing Fault Model on a simple circuit.

Sampling Fault Model To the best of our knowledge, the most recent description of the Sampling Fault Model is found in [9]. This article explains that an EM-pulse induces parasitic currents in wire loops located beneath the probe. Consequently, fluctuations occur in the current of affected wires, such as `Vdd`, `Gnd`, the clock tree, DFF routing, and others, causing voltage bounces and drops, depending on the injection polarity.

In scenarios where a voltage drop occurs, the pulse causes circuit signals ($\texttt{D}_1$, $\texttt{Q}_1$, and `clk` in Figure 3) to temporarily decrease, halting circuit operation. These signals eventually return to their nominal values. If the injection occurs just before a rising clock edge (as depicted in Figure 3), a race ensues between D$_1$ and clk. A fault occurs if clk returns to its nominal value before D$_1$.

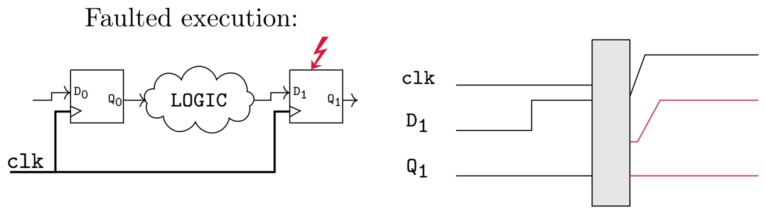


Fig. 3: Sampling Fault Model on a simple circuit. The grey rectangle in the chronogram symbolizes the signal drop.

This model has been further explored and reproduced in [27], with refinements to the coupling model and coverage of cases not previously examined: how to create a fault with a positive pulse when the DFF input is high. A distinctive characteristic of the Sampling Fault Model is the fault sensitivity window which has been experimentally confirmed: a specific timing window relative to the rising edge of the clock, during which faults can be induced. Because of the race condition between clk and D$_1$, a fault can only occur within a narrowly defined timing window around the clock's rising edge. This sensitivity window remains constant for a given circuit, irrespective of its frequency, but is influenced by the logic situated between the DFFs.

What about the Charge-Based Fault Model? Another model, described by Liao *et al.* [13], is the Charge-based Fault Model. This model posits that an EM-pulse influences the circuit's capacitance. When a circuit is overclocked or powered with subnominal voltage, the amount of charge present is closer to the threshold required to flip a DFF under normal conditions. As a result, EM-pulses can more easily perturb DFFs. Within the scope of this paper, this model is not considered. While it presents an interesting concept, it is primarily supported by experimental data; no comprehensive explanatory model or simulation has yet been proposed. The Charge-Based Fault Model does not provide a clear explanation of why a DFF would store erroneous data, it only states that charges influence this outcome. Instead, it can be viewed as a description of how other factors might aid in the facilitation of EMFI.

Comparison with Controlled Synchronous Clock Glitch To develop a fault model that explains why the CSCG causes faults, it's essential to identify which component is most susceptible to being impacted. Given that the glitch is carried out by the clock, we hypothesize that DFFs are likely to be affected, as illustrated in Figure 4.

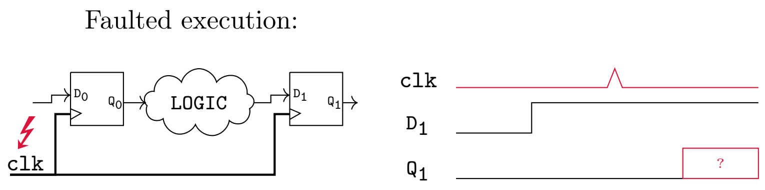


Fig. 4: Controlled synchronous clock glitch impact on a simple circuit.

Furthermore, the Timing Fault Model and the Sampling Fault Model do not sufficiently explain why the CSCG causes faults since:

– Only the clock is modified and other signals, particularly $\mathtt{D}_1$, remain stable, there is no race condition between `clk` and $\mathtt{D}_1$. Therefore, the Sampling Fault Model is excluded.
– There are no timing variations for either the clock or $\mathtt{D}_1$, thus no setup time violations can occur, ruling out the Timing Fault Model.

Given that the CSCG cannot be accounted for by existing published fault models, further study is necessary to identify a fault model that accurately describes its effects.

3 Understanding the Controlled Synchronous Clock Glitch

Considering the published fault models introduced in Section 2, the observed fault model needs to be thoroughly analyzed. In this section, we propose several hypotheses that can explain the CSCG fault model.

Hypothesis 1 (Energy Threshold) *For a DFF to correctly sample a clock's rising edge, the clock signal must meet a certain energy threshold, combination of voltage amplitude and width thresholds.*

The energy of the clock signal determines whether the DFF samples the incoming data. A failure to sample is considered a fault. Depending on the energy of the clock signal, three states of the DFF are observed:

1. When the energy of the clock signal is too low, falling below the required energy threshold, the DFF is in a *always faulted* state.
2. Conversely, when the energy of the clock signal is sufficiently high, surpassing the threshold, the DFF is in an *always unfaulted* state.
3. When the clock signal hovers around the required threshold, the DFF enters a *sometimes unfaulted* state (i.e. when out of `X` FIs, it has sampled at least once). In this state, the output of the DFF is in a metastable state, influenced by the amount of clock energy. This phenomenon is further explored in 5.1.

This hypothesis alone is insufficient to fully explain the effect of the CSCG. We propose below two additional hypotheses, following the introduction of the *fault sensitivity* concept.

Definition 1 (Fault Sensitivity). *The minimum amplitude at which a DFF becomes sometimes unfaulted is called its fault sensitivity.*

When faulting two DFFs, for instance on a field programmable gate array (FPGA), their behaviors should be similar but not identical, and they may not share the same *fault sensitivity*. This difference can be attributed to variability in the manufacturing process among integrated circuits (ICs) and within individual DFFs of the same IC die. In other words, two identical DFFs, i.e., with the same characteristics and hardware layout, may not share the same *fault sensitivity*. Also, if the two DFFs are from two FPGAs of the same model, they may not share the same *fault sensitivity*.

Hypothesis 2 (Fault Sensitivity Dependency on Intrinsic Properties) *The fault sensitivity of a DFF depends on its intrinsic properties, such as process variability and clock routing up to the DFF among others.*

However, the intrinsic properties alone are not sufficient to explain observed variations in fault sensitivity. To add a layer of complexity, we consider the environment surrounding the glitched DFFs, specifically focusing on the wires carrying signals (e.g., `clock`, `Vdd`, `Vss` between DFFs). The energy from neighboring wires may influence the glitched clock, altering the behavior of the target DFFs. This includes both data routing between DFFs and the clock routing on the dedicated clock network.

Hypothesis 3 (Fault Sensitivity Dependency on Extrinsic Properties) *The fault sensitivity of a DFF may also be affected by extrinsic factors, such as the activity in neighboring wires (including routing between DFFs and the routing of the clock tree).*

To validate these hypotheses, experiments (either through simulation or on actual hardware) are necessary. In the following sections, the experimental setup is presented.

4 Experimental Setup

The previous section has introduced hypotheses that may explain the effects of the CSCG. In this section, we describe experiments aimed at confirming or refuting these hypotheses. The experimentation is categorized into two types: physical FI and simulated experiments.

4.1 Physical Experiments

Physical FIs are performed using TRAITOR implemented on an Artix-7 FPGA to inject CSCG into our device under test (DUT). To facilitate comprehension, we will begin by elucidating the use of TRAITOR for FI, embedded into our DUT.

In preparation for subsequent discussions, it is imperative to make a clear distinction between logical DFF and physical DFF:

– The logical DFF represents an abstract conceptualization of a DFF in our DUT, with multiple possible mappings onto physical DFFs.
– The Physical DFFs are tangible components found on the ICs, such as FPGAs, serving as the foundational element for logical DFF. A logical DFF is mapped onto a given physical DFF.

When logical or physical is not mentionned, then the representation of the DFFs can be either.

How does TRAITOR work? To generate a CSCG, we can control the occurrence of the corrupted clock edge in each clock cycle and adjust a single parameter known as the **amplitude**, which shapes the corrupted edge. Figure 5 illustrates the generation of the corrupted edge using two phase-shifted clocks, with the phase under the TRAITOR user's control. Ideally, this method would result in a square pulse. However, the theoretical pulse width, equivalent to the phase shift, is too small relative to the circuit's inductance, preventing the signal from reaching its high value within the available time. Consequently, the **amplitude** of the corrupted edge is determined by the phase shift; a larger phase shift allows the corrupted signal to reach a higher level. Thus, the amplitude parameter influences both the height and the duration (also referred to as width) of the corrupted clock edge.

In our implementation, the phase shift is adjustable in increments of 32 ps. Throughout this paper, the term 'amplitude' applied to TRAITOR will refer to the number of these 32 ps steps in the phase shift.

TRAITOR produces two clocks: a regular clock, referred to as `clk_ok`, and a clock that incorporates the CSCG, referred to as `clk_glitched`. Both clocks are synchronous, operating at 16MHz, and are supplied to the DUT.

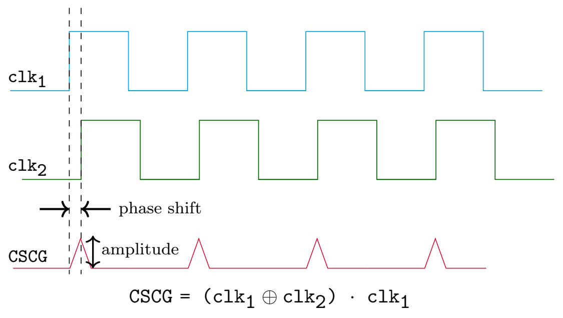


Fig. 5: The CSCG is generated using two out-of-phase clocks, `clk1` and `clk2`. The TRAITOR user has the capability to replace the regular clock signal with CSCG at their discretion.

Composition of the DUT The DUT, depicted in Figure 6, comprises several logical DFFs, categorized into two types:

1. Target logical DFFs that receive `clk_glitched`.
2. Control logical DFFs that receive `clk_ok`.

These DFFs are organized into groups of 6, with a group consisting of either target logical DFFs (referred to as a target chain) or control logical DFFs (referred to as a control chain). There is one control chain and 32 target chains. Each chain, whether control or target, is fed the same input: a sequence alternating between 0 and 1. This sequence ensures that the content of every DFF, whether logical or physical, changes with each clock cycle. The outputs of the target chains are compared with the output of the control chain. Any discrepancy in at least one target output is indicative of a fault. By examining the timing between the FI and the appearance of the faulty output at the end of a chain, the specific logical DFF affected in the chain can be identified.

The logical DFFs are mapped onto the physical DFFs of an Artix-7 FPGA, which are located in slices (8 physical DFFs per slice). Although slices contain other components, for clarity these are not considered in our discussion. Two distinct mappings, as shown in Figure 7, are used to investigate how these mappings influence our results.

Fault Injection Protocol The physical experiments are detailed in Section 5. To conduct these experiments, the following protocol is adhered to:

1. Both TRAITOR and the DUT are implemented on the same Artix-7 FPGA. This setup ensures the shortest and simplest clock paths, avoiding additional hardware components such as IOs or external wiring. To ensure consistency across all experiments, we meticulously determine the placement of

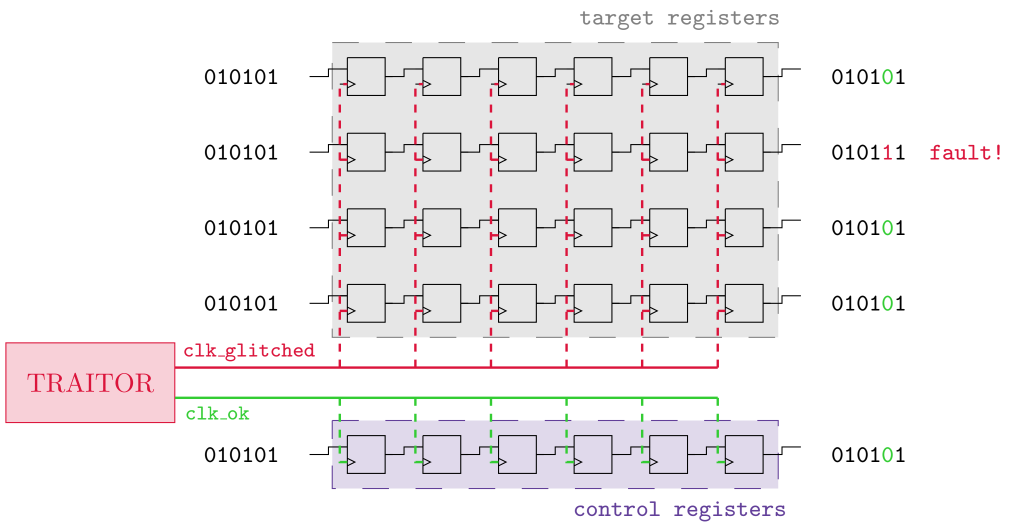


Fig. 6: DUT and TRAITOR on an Artix-7 FPGA.

TRAITOR and our DUT on the FPGA, aiming for precision. This guarantees that TRAITOR is consistently mapped to the same location for every experiment.

2. The FI process remains constant. Upon receiving a trigger from the target, a CSCG with a specified amplitude is injected. This process is repeated 100 times for each amplitude, ranging from 0 to 29. Subsequently, we analyze which DFF, if any, has been impacted by the FI.
3. Conclusions are drawn based on the observed outcomes and the analysis of results.

4.2 Transistor-Level Simulations

The simulations were carried out using Eldo [2], an ASIC oriented SPICE simulator. Given the proprietary nature of the Artix-7 FPGA design, replicating the exact 28 nm physical DFFs targeted in the physical experiments is not feasible. Instead, the simulations employ DFFs from a similar CMOS technology available in our laboratory, i.e., a 28 nm FDSOI (Fully Depleted Silicon On Insulator) Process Design Kit. These physical DFFs feature three connections: `D`, `Q`, and a clock input. However, unlike the physical DFFs used in the Artix-7 experiments, they lack a reset pin. Although the simulated DFFs and the Artix-7 physical DFFs do not have the same implementation, they do not significantly differ since they are designed for similar technology node and tend to behave the same way.

The simulated circuit consists of two DFFs. They first undergo a normal clock cycle, followed by a glitched one. Although a fault is injected into both

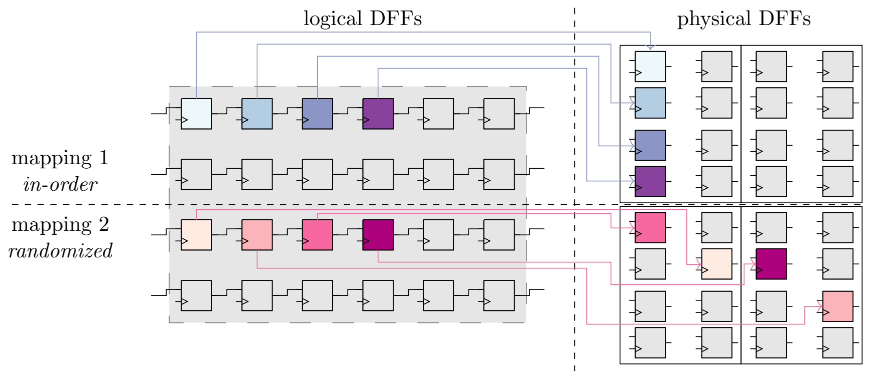


Fig. 7: The two logical-to-hardware mappings: mapping 1 is in-order and mapping 2 is randomized.

DFFs, only the first one is considered for analysis; the second DFF is included to more closely mirror our physical experiments by simulating a load. The clock operates at 100 MHz with a voltage amplitude ranging from 0 V to 1 V.

The simulation focuses on a state change in the first DFF, transitioning from 0 to 1. It is important to note that the metastability phenomenon observed in physical experiments is not replicable in simulation. The primary goal of the simulation is to estimate the impact of the voltage and width of the controlled synchronous clock glitch. To achieve this, we independently vary both parameters, incrementally increasing them from low values until the DFF under test samples the input.

5 Hypotheses validation

In this section, multiple experiments and simulations are conducted to validate the hypotheses presented in Section 3.

5.1 Hypothesis 1: Energy Threshold

We examine the behavior of physical DFFs faulted with TRAITOR, observing variations depending on the amplitude parameter. The results of the FI campaign validate Hypothesis 1. The target physical DFFs exhibit the following behaviour, shown on Figure 8, for 3 distinct DFFs:

1. For amplitudes ranging from 0 to 21, inclusive, all DFFs are in a *always faulted* state.
2. For amplitudes between 22 and 24, inclusive, some DFFs are in a *always faulted* state, while others are in a *sometimes unfaulted* state.
3. Starting from amplitude 25, all DFFs are in a *always unfaulted* state.

The energy threshold is not identified by a single amplitude; instead, it is characterized by a range of 2 to 3 amplitudes in this experiment, with the fault sensitivity as the lower bound. During this transition phase from faulted to unfaulted, a physical DFF progressively experiences fewer faults until it becomes entirely unfaulted. The transition phases of the 192 physical DFFs overlap but are not identical, as illustrated in Figure 8.

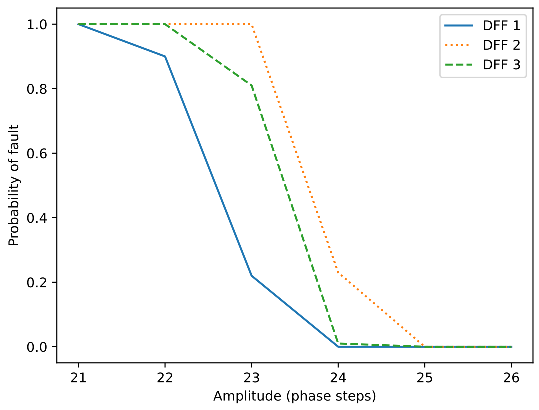


Fig. 8: Transitions phases of three target physical DFFs chosen since they exhibit different characteristics.

Energy Propagation and Metastability Figure 8 illustrates the variation in fault occurrence probability relative to the glitch amplitude for 3 distinct DFFs, selected for their different characteristics. This figure shows various behaviors.

First, the 3 DFFs exhibit different fault sensitivities (22 for DFF 1, 23 for DFF 3, 24 for DFF 2). DFF 2 remains in a *always faulted* state at a higher amplitude, suggesting more energy loss during clock signal propagation. The causes of this energy loss are examined with Hypotheses 2 and 3. Then, at an amplitude equal to their fault sensitivity, each DFF shows a fraction of samplings being unfaulted and the rest faulted, indicative of metastable behavior. This ratio is consistent and reproducible for each physical DFF.

The standard error of the mean (SEM) is easily calculable: in the worst-case scenario where the fault probability is $p = 0.5$, the standard deviation σ is $\sigma = \sqrt{p \cdot (1-p)} = 0.5$. Therefore, the SEM is $SEM = \sigma/100 = 0.005$, as we have 100 experiments for each DFF. We can deduce that our fault probability falls within 3 error deviations (= 0.015) for approximately 99 % confidence. For instance, our metastability evaluation suggests that at amplitude 23, DFF 1 registers a fault in $22 \pm 1.5\%$ of injection attempts with 99 % confidence.

What we observe in Figure 8 is a typical S-curve characteristic of metastable behavior due to insufficient energy at the DFF's clock pin [4]. However, only one amplitude per DFF triggers the metastable output.

As a conclusion, each DFF undergoes a transition phase, displaying a limited metastable behavior. The transition phases of different DFFs may overlap but are not identical, attributed to the energy loss during clock signal propagation. This results in a collective transition phase for all DFFs from amplitudes 22 to 24 inclusive.

Simulating the Influence of Glitch Width and Voltage Amplitude Independently While the previous experiment emphasizes the existence of an energy threshold, TRAITOR's specific design does not allow for independent testing of the influence of the glitch width and voltage amplitude on this threshold. To overcome this limitation, we simulate a small circuit (as described in Section 4.2) where we send a glitched clock pulse while varying the glitch width and voltage amplitude independently to observe if the sampling occurs.

The simulation is performed with a glitch width ranging from 0 ns to 5 ns by steps of 0.01 ns and voltage amplitude ranging from 0 V to 1 V by steps of 0.01 V). Figure 9 illustrates the sampling behavior with respect to glitch width and voltage amplitude parameters. The DFF successfully samples above the curve. The plot is constrained to the range from 0 ns to 1 ns, reflecting the fact that the amplitude reaches a lower plateau at 0.46 V.

Remarkably, sampling occurs for very small widths, as long as the voltage amplitude is sufficiently high; the minimum width for this occurrence is 0.03 ns at a voltage of 0.84 V. However, the opposite is not true: the glitch voltage amplitude must be at least 0.46 V for the DFF to sample, regardless of the width. In other words, it is "sufficient" for the DFF to sample that the controlled synchronous clock glitch has a high voltage amplitude for a short width, but not a long width with a low amplitude. Hence, the voltage amplitude threshold appears to be more restrictive than the width threshold.

5.2 Hypothesis 2: Fault Sensitivity Dependency on Intrinsic Properties

In this part of the study, we aim to understand why the transition phase, particularly the fault sensitivity, varies among physical DFF. Our primary focus is on the potential dependency of fault sensitivity on the intrinsic properties of physical DFF.

As discussed in Section 5.1, each physical DFF exhibits a specific and reproducible fault sensitivity. One primary factor influencing this sensitivity is the layout of clock routing: not all physical DFF on an FPGA share identical clock signal paths. Variations in these paths, potentially due to length differences or coupling with neighboring wires, can introduce disparities in inductance. If the layout of the clock routing was the sole intrinsic factor affecting fault sensitivity, then replicating the same design (with identical mapping) on another FPGA of the same model would result in the same sensitivity for identical logical DFFs.

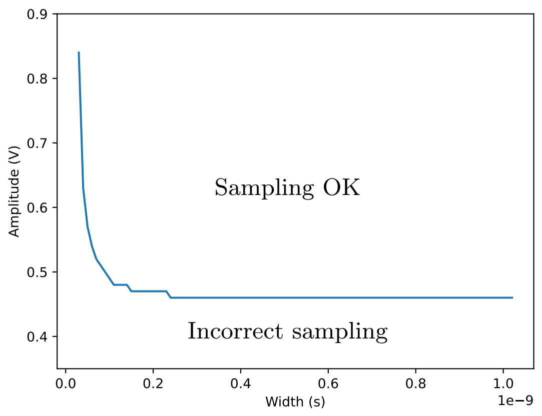
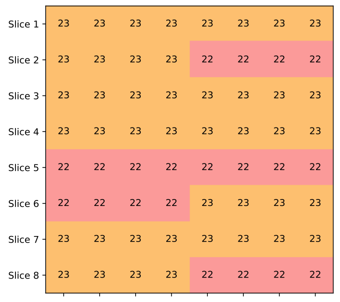


Fig. 9: Simulated sampling results: for a given glitch with voltage amplitude and width above this curve, sampling is correct.

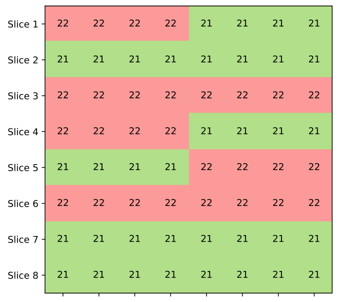
In the ensuing experiment, the same DUT is mapped onto two Artix-7 FPGAs in the exact same manner. Practically, this involves using the same bitstream FPGA image on both FPGAs. The resulting fault sensitivities are depicted in Figures 10a and 10b. One can see that while the fault sensitivities of the two FPGAs do show some similarities, notable differences exist. Given that both FPGAs are programmed with the same image and therefore have identical clock routings, the discrepancies observed in Figure 10 can be attributed to process variations. The individual FPGA dies are not exactly identical, leading to variations in the inductances of clock paths, which in turn result in differing fault sensitivities for placement-equivalent physical DFFs.

Limits to the Intrinsic-Only Fault Model If we assume that only the intrinsic properties of a physical DFF affect its fault sensitivity, then the mapping of logical DFFs to physical DFFs on a specific FPGA should not influence the fault sensitivities of these physical DFFs. This is because the clock routing is independent of the routing of other signals. Since the glitch is propagated solely by the clock signal, the fault sensitivity, assuming it depends solely on intrinsic properties, would be specific to each physical DFF.

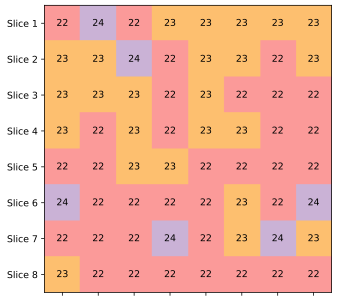
To test this assumption, we map the same physical DFFs onto the same FPGA in two different configurations (as depicted in the two mappings of Figures 10a and 10c) and then compare their fault sensitivities. This results in varying fault sensitivities. The `clk_glitched` signal remains consistent across both mappings since it follows dedicated clock paths, suggesting that the CSCG should be identical in both experiments and consequently result in the same fault sensitivity for each physical DFF independently from the mapping. The two map-



(a) Color coded fault sensitivities of the first 64 registers on mapping 1 *in-order* on FPGA 1.



(b) Color coded fault sensitivities of the first 64 registers on mapping 1 *in-order* on FPGA 2.



(c) Color coded fault sensitivities of the first 64 registers on mapping 2 *randomized* on FPGA 1.

Fig. 10: Comparing fault sensitivities between physical DFFs for various settings.

pings differ in how data signals are routed between physical DFFs which clearly has an impact on the fault sensitivity. This observation leads us to hypothesize that extrinsic properties, such as data signals in this case, may influence CSCG.

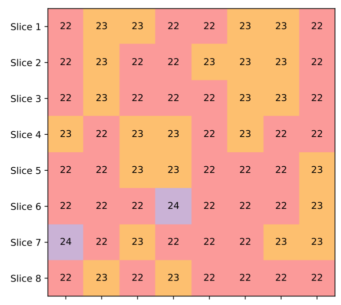
5.3 Hypothesis 3: Fault Sensitivity Dependency on Extrinsic Properties

Given that intrinsic properties alone do not account for all variations in fault sensitivity, we now turn our attention to extrinsic properties. Specifically, we examine two types of extrinsic influences:

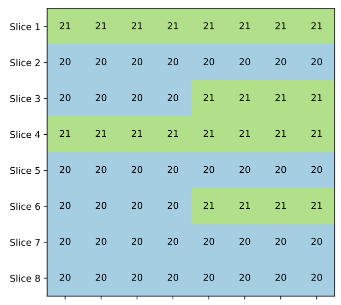
– Activity on data wires, i.e., the paths linking physical DFFs to each other.
– Activity on clock wires, responsible for carrying clock signals to the physical DFFs.

For each wire type, we conduct a separate experiment to isolate and observe its specific influence.

Impact of Data Wires The influence of data wires on the `clk_glitched` energy was previously suggested in Section 5.2. We delve deeper into this aspect with the following experiment. Figures 10a and 11a illustrate the fault sensitivities of two different routings. Similar to the approach in Section 5.1, the target logical DFFs are mapped *in-order*. However, in this case, we alter the routing between two physical DFFs, resulting in a change in the data wire connections between them. Consequently, we end up with two DUT having the same logical DFFs to physical DFFs mapping, and thus identical clock routing, but differing in data wire routing between physical DFFs.



(a) Color-coded fault sensitivities of the first 64 registers on mapping 1 *in-order* with different data routing on FPGA 1, to be compared to Figure 10a.



(b) Color-coded fault sensitivities of the first 64 registers on mapping 1 *in-order* with a forced adjacent path for the clock on FPGA 1, to be compared to Figure 10a.

Fig. 11: Comparing fault sensitivities between physical DFFs for different routing

As a conclusion, the routing of data signals between physical DFFs significantly impacts the energy of the clock signal reaching these physical DFFs, thereby affecting their fault sensitivities.

Impact of Clock Wires We further hypothesize that the `clk_glitched` signal is influenced by the proximity to the `clk_ok` signal. Previously, the mapping of the DUT and TRAITOR was carefully arranged to avoid any crossing or parallel arrangement of the two clock networks. To assess the impact of clock network interference, we now map some control DFFs on a slice adjacent to the target DFFs. This setup places both `clk_glitched` and `clk_ok` on parallel physical paths, given that the dedicated clock routes are next to each other and originate from nearby sources [1].

Figures 10a and 11b show that the fault sensitivities not only differ but are also notably lower. In all previous experiments on this FPGA, such as on Figure 10a, fault sensitivities ranged between 22 and 24. However, in this setup, they range between 20 and 21. It appears that positioning the `clk_ok` signal adjacent to `clk_glitched` effectively 'adds energy' to the latter, thereby reducing its fault sensitivity.

Interpretation The observed energy transfers, both data-to-clock and clock-to-clock, are likely the result of cross-talk between these signals. The fault sensitivity of a physical DFF is highly dependent on the energy delivered by the clock's rising edge, so even a small amount of energy added or subtracted through cross-talk can have a noticeable impact [11]. These findings suggest that the fault sensitivity in a DFF is an extremely precise indicator of the activity in the surrounding circuitry.

6 Conclusion

EMFI is a popular yet imprecise method for inducing incorrect behaviors in ICs. Among the various effects caused by EMFI, it triggers a specific clock glitch known as the synchronous clock glitch. We reproduce a similar signal, referred to as the CSCG, directly on the clock network, delivering less energy than a regular clock cycle. This glitch results in faults, but existing physical fault models have not adequately explained it thus far.

To address this knowledge gap, we propose the Energy-threshold Fault Model. This model states that a DFF requires a specific energy level on the clock port to sample correctly. When the energy falls below the required threshold, the DFF fails to sample. When the energy hovers near that threshold, the DFF output enters a metastable state, leading to uncertain sampling. The threshold of each DFF varies based on intrinsic properties, such as process variability and clock network layout. Moreover, it can be influenced by extrinsic factors like the activity of neighboring wires, due to cross-talk. This suggests that measuring the threshold value provides a novel and highly precise means of assessing the activity of a circuit surrounding a specific DFF.

It is worth noting that this model was only tested using the same family of FPGAs. While the relationship between clock energy and sampling should hold true for any IC (as suggested by simulations in Section 4.2), the specifics of the cross-talk phenomenon might vary. Additionally, our testing was limited to the use of TRAITOR; future work should focus on recreating CSCG with EMFI as Claudepierre *et al.* [5] and verify if the Energy-threshold Fault Model requires adjustments.

Bibliography

[1] 7 Series FPGAs Clocking Resources. https://docs.xilinx.com/v/u/en-US/ug472_7Series_Clocking.

[2] Eldo Platform. https://eda.sw.siemens.com/en-US/ic/eldo/.

[3] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. When Clocks Fail: On Critical Paths and Clock Faults. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010, Passau, Germany. Proceedings*, volume 6035 of *Lecture Notes in Computer Science*, pages 182–193. Springer, April 2010.

[4] Doris Chen, Deshanand Singh, Jeffrey Chromczak, David Lewis, Ryan Fung, David Neto, and Vaughn Betz. A Comprehensive Approach to Modeling, Characterizing and Optimizing for Metastability in FPGAs. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, page 167–176, February 2010.

[5] Ludovic Claudepierre and Philippe Besnier. Microcontroller Sensitivity to Fault-Injection Induced by Near-Field Electromagnetic Interference. In *APEMC - Asia-Pacific International Symposium on Electromagnetic Compatibility*, pages 1–4, Sapporo, Japan, June 2019.

[6] Ludovic Claudepierre, Pierre-Yves Péneau, Damien Hardy, and Erven Rohou. TRAITOR: A Low-Cost Evaluation Platform for Multifault Injection. In Weizhi Meng and Li Li, editors, *ASSS'21: Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems, Virtual Event, Hong Kong*, pages 51–56. ACM, June 2021.

[7] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES. In Guido Bertoni and Benedikt Gierlichs, editors, *Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium*, pages 7–15. IEEE Computer Society, September 2012.

[8] Mathieu Dumont, Mathieu Lisart, and Philippe Maurine. Electromagnetic Fault Injection: How Faults Occur. In *Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2019, Atlanta, GA, USA*, pages 9–16. IEEE, August 2019.

[9] 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, 2021.

[10] Antoine Gicquel, Damien Hardy, Karine Heydemann, and Erven Rohou. SAMVA: Static Analysis for Multi-fault Attack Paths Determination. In Elif Bilge Kavun and Michael Pehl, editors, *Constructive Side-Channel Analysis and Secure Design - 14th International Workshop, COSADE 2023,*

Munich, Germany, Proceedings, volume 13979 of *Lecture Notes in Computer Science*, pages 3–22. Springer, April 2023.

[11] Ilias Giechaskiel, Kasper B. Rasmussen, and Ken Eguro. Leaky wires: Information leakage and covert communication between fpga long wires. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, page 15–27, New York, NY, USA, 2018. Association for Computing Machinery.

[12] Vanthanh Khuat, Jean-Luc Danger, and Jean-Max Dutertre. Laser Fault Injection in a 32-bit Microcontroller: from the Flash Interface to the Execution Pipeline. In *18th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2021, Milan, Italy*, pages 74–85. IEEE, sep 2021.

[13] Haohao Liao and Catherine H. Gebotys. Methodology for EM Fault Injection: Charge-based Fault Model. In Jürgen Teich and Franco Fummi, editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy*, pages 256–259. IEEE, March 2019.

[14] Philippe Maurine. Techniques for EM Fault Injection: Equipments and Experimental Results. In Guido Bertoni and Benedikt Gierlichs, editors, *Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium*, pages 3–4. IEEE Computer Society, sep 2012.

[15] Roukoz Nabhan, Jean-Max Dutertre, Jean-Baptiste Rigaud, Jean-Luc Danger, and Laurent Sauvage. Highlighting two em fault models while analyzing a digital sensor limitations. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–2, 2023.

[16] Sebastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine. Electromagnetic fault injection: the curse of flip-flops. *Journal of Cryptographic Engineering*, 7(3):183–197, 2017.

[17] Pierre-Yves Péneau, Ludovic Claudepierre, Damien Hardy, and Erven Rohou. NOP-Oriented Programming: Should we Care? In *IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2020, Genoa, Italy*, pages 694–703. IEEE, September 2020.

[18] Cyril Roscian, Jean-Max Dutertre, and Assia Tria. Frontside laser fault injection on cryptosystems - application to the aes' last round -. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 119–124, 2013.

[19] Nidhal Selmane, Sylvain Guilley, and Jean-Luc Danger. Practical Setup Time Violation Attacks on AES. In *Seventh European Dependable Computing Conference, EDCC-7 2008, Kaunas, Lithuania*, pages 91–96. IEEE Computer Society, May 2008.

[20] Niek Timmers, Albert Spruyt, and Marc Witteman. Controlling PC on ARM Using Fault Injection. In *Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA*, pages 25–35. IEEE Computer Society, aug 2016.

[21] Simon Tollec, Mihail Asavoae, Damien Couroussé, Karine Heydemann, and Mathieu Jan. Exploration of Fault Effects on Formal RISC-V Microarchitecture Models. In *Workshop on Fault Detection and Tolerance in*

Cryptography, FDTC 2022, Virtual Event / Italy, pages 73–83. IEEE, sep 2022.

[22] Thomas Trouchkine, Guillaume Bouffard, and Jessy Clédière. Fault Injection Characterization on Modern CPUs. In Maryline Laurent and Thanassis Giannetsos, editors, *Information Security Theory and Practice - 13th IFIP WG 11.2 International Conference, WISTP 2019, Paris, France, Proceedings*, volume 12024 of *Lecture Notes in Computer Science*, pages 123–138. Springer, December 2019.

[23] Thomas Trouchkine, Guillaume Bouffard, and Jessy Clédière. EM Fault Model Characterization on SoCs: From Different Architectures to the Same Fault Model. In *18th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2021, Milan, Italy*, pages 31–38. IEEE, sep 2021.

[24] Thomas Trouchkine, Sébanjila Kevin Bukasa, Mathieu Escouteloup, Ronan Lashermes, and Guillaume Bouffard. Electromagnetic fault injection against a complex CPU, toward new micro-architectural fault models. *Journal of Cryptographic Engineering*, 11(4):353–367, 2021.

[25] Shih-Yi Yuan, Yu-Lun Wu, Richard Perdriau, Shry-Sann Liao, and Hao-Ping Ho. Electromagnetic interference analysis using an embedded phase-lock loop. In *Asia-Pacific Symposium on Electromagnetic Compatibility*, pages 189–192, 2012.

[26] Bilgiday Yuce, Patrick Schaumont, and Marc Witteman. Fault Attacks on Secure Embedded Software: Threats, Design, and Evaluation. *Journal of Hardware and Systems Security*, 2(2):111–130, 2018.

[27] Maoshen Zhang, He Li, and Qiang Liu. Deep Exploration on Fault Model of Electromagnetic Pulse Attack. *IEEE Transactions on Nanotechnology*, 21:598–605, 2022.