# MUX-based Polymorphic Registers and FSMs to Protect Roots of Trust from Voltage Fault Injection

Sourav Roy
*Electrical and Computer Engineering (ECE)*
*University of Florida, USA*
sourav.roy@ufl.edu

Domenic Forte
*Electrical and Computer Engineering (ECE)*
*University of Florida, USA*
dforte@ece.ufl.edu

***Abstract*—Modern electronic systems such as FPGAs, processors and SoCs are equipped with roots of trust (RoT) to ensure confidentiality, availability and integrity. However, malicious parties can carry out non-invasive voltage fault injection (VFI) attacks to break the RoT. VFI or voltage glitch attack is powerful enough to break security of modern processors from top vendors such as Arm, Intel, AMD etc. Existing voltage glitch detectors require separate response mechanism, and the latency between detection and response make them ineffective. In an effort to integrate detection and response in one countermeasure, recently a NAND/NOR-based polymorphic latch was introduced which changes its behavior with supply voltage. In this work, multiplexer (MUX)-based polymorphic latches, registers, and FSMs are designed and implemented in cryptographic benchmarks capable of destroying data within about $1ns$ of attack initiation which is about $80\times$ improvement compared to the previous NAND/NOR-based designs.**

***Index Terms*—Voltage fault injection, dynamic voltage and frequency scaling, polymorphic latch, polymorphic register, polymorphic FSM.**

## I. Introduction

The hallmark of a secure computing system is its root of trust (RoT) which is a hardware-based security module embedded within a processor or system-on-chip (SoC) that securely stores cryptographic keys, enforces secure boot, and enables trusted execution. It ensures the integrity, authenticity, and confidentiality of system operations by protecting sensitive data, managing encryption and decryption, and preventing unauthorized access or extraction. Confidentiality is achieved by encryption cores such as advanced encryption standard (AES), data encryption standard (DES), Rivest Shamir Adleman (RSA), etc. Integrity refers to ensuring that information (data and code) remains unaltered, accurate, and consistent. In modern integrated chips (ICs), it is often accomplished by authenticity checks using password or digital signature verification. During the system boot, an authenticity check is done and only the verified applications are allowed to load. This process is referred to as "secure boot". Some examples of secure computation platforms are Arm TrustZone, Intel secure guard extension (SGX), AMD secure processor (SP), etc. These are essentially RoTs in modern SoCs.

In modern ICs, voltage and frequency can be lowered during light workloads to save power. Such mechanisms have been leveraged to carry out voltage fault injection (VFI) attacks on the above-mentioned secure platforms via software [1], [2] or with physical access to the voltage pin/regulator [3] to break confidentiality and/or integrity. Attackers have injected faults to steal cryptographic keys, stall computation indefinitely [2], and bypass security boot to load untrusted malicious applications [4] as shown in Fig. 1. Existing Voltage glitch detectors present on-chip are often too slow to stop such attacks as they require separate response mechanism. Recently, a NAND/NOR-based polymorphic latch [5] was proposed which integrates detection and response which destroyed the stored data within $84ns$ of initiation of VFI. Although this countermeasure can protect cryptographic keys from being stolen by VFI attacks, it cannot address the VFI attack on control FSMs. It also has relatively large area, power, and timing overhead.

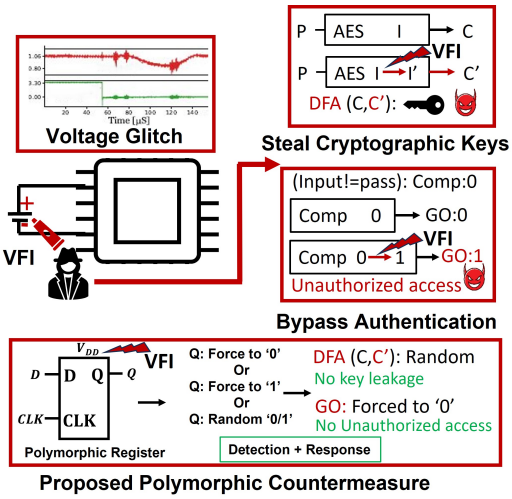



Fig. 1: Illustration of VFI undervolting the supply line, attack outcomes of VFI, and proposed polymorphic countermeasure integrating detection and response.

In this paper, multiplexer (MUX)-based polymorphic latches, registers and FSMs are designed which are simpler in design, smaller, faster and power-efficient compared to their NAND/NOR-based counterparts. The proposed MUX-based polymorphic latches and registers operate as normal memory elements under normal operating voltage when there is no attack. During VFI attack, whenever the supply voltage is lowered below the designer's desired threshold, the polymorphic behavior kicks in and the data stored in polymorphic latches and registers gets changed within a few nanoseconds as shown in Fig. 1. The threshold for polymorphism can be controlled by careful transistor sizing using genetic algorithm. Contributions of this work are summarized as follows:

- Design of multiplexer (MUX)-based polymorphic latches and registers and FSMs to protect chips from VFI attacks in a simpler, faster and more efficient way compared to NAND/NOR-based counterparts.
- Implementation of MUX-based polymorphic registers in three cryptographic benchmarks: AES, DES and RSA to compare area, power, timing overhead, reliability, and (intended) corruptibility.

The rest of the paper is organized as follows. Section II provides background on VFI and countermeasures. In Section III, three different MUX-based polymorphic latch/register designs and operations are introduced. Section IV describes the implementation of polymorphic FSM utilizing the proposed registers. In Section V, power, perfor-

TABLE I: Impact of VFI attacks in terms of capability, cost and time needed to carry out the attacks.

| | Breach | Root-of-trust | Time Needed | Cost |
|---|---|---|---|---|
| [1] | Confidentiality | TrustZone | 2 $days$ | \$0 |
| [2] | Confidentiality, Availability | Intel SGX | 2 $hrs$ | \$0 |
| [3] | Confidentiality | Intel SGX | 2 $mins$ | \$30 |
| [4] | Confidentiality, Integrity | AMD SP SEV | 46 $mins$ | $<$ \$100 |

mance, area (PPA), and reliability characterizations of polymorphic latches and registers are presented along with benchmark simulations. Finally, Section VI concludes this paper.

## II. Background

While there are many approaches to fault injection, VFI attacks are often considered the most popular and inexpensive type. During VFI attacks, an attacker initiates undervoltage either via software (leveraging access to the DVFS mechanism) or via hardware using access to the chip's power supply rail. These attacks can be optimized to bypass the voltage monitoring sensors on-chip.

### A. VFI Attack Effectiveness and Threat Model

Noninvasive attacks such as VFI have compromised even trusted execution environments (TEEs) such as Arm TrustZone, Intel SGX, AMD SP which are specially designed with security in mind. Through a software-based framework, faults can be injected into a circuit at the appropriate time, and the fault results can be analyzed using software designed to extract assets through analysis of circuit behavior under faults. In [1], voltage glitches were used to bypass the TrustZone. A similar vulnerability in the DVFS of Intel processors was discovered in [2]. A physical hardware-based VFI approach was later adopted [3] to undermine any software patching. Physical VFI was used to break the security of AMD processors with SP capability [4]. The impact of these attacks against commercial chips is outlined in Table I. One of the most concerning aspects is that, the software-based attacks can be carried out without any extra cost and even the hardware-based attacks do not cost more than 100\$. Not only that, the attacks can be carried out within a matter of minutes after benchmarking and voltage glitching preparations are done. Such capability of undermining state-of-the-art RoTs and TEEs of modern chips make this kind of attack extremely potent and dangerous.

Voltage glitch attacks occur outside the normal range of IR drops and dynamic voltage droop seen during regular chip operation. The change in voltage magnitude is often several hundred millivolts up to a volt. The voltage glitch duration is usually tens of microseconds but can be as fast as few hundred nanoseconds [6]. Thus, a countermeasure is required which is fast enough to react within tens of nanoseconds before the voltage glitch subsides.

### B. Limitation of Existing Countermeasures

Voltage glitch detectors or probe detectors that detect sudden voltage fluctuations in the supply line are available as countermeasures against VFI attacks. For detecting environmental disturbances, ring oscillators (ROs) can be used to measure frequency mismatch [7] but ROs have high area and power overhead and suffer from reliability issues over time. Other circuit-based detectors may be able to detect voltage modulation, but not voltage lowering or sudden voltage glitches, such as clock freeze and voltage modulation detectors reported in [8]. Other voltage manipulation detectors are available such as on-chip monitors (OCMs) [9]. Integrating the detection and response in one circuitry, a self-destructive polymorphic latch [5] was recently proposed. The polymorphic latch was designed using polymorphic NAND/NOR gate which changes behavior from NOR to NAND under attack condition and destroys the data stored in the latch. The countermeasures are categorized in Table II according to type of attacks and nature of the countermeasure in terms of detection and response mechanism. Response mechanisms such as self-destruction [10] and zeroization [11] requires separate detection mechanism. The clear advantage of proposed MUX-based countermeasure over the existing ones is the integration of detection and response in a simpler, faster way with less overhead.

TABLE II: Existing Countermeasures. I: Invasive attack, SI: Semi-invasive attack, NI: Non-invasive attack, S: Sense-based, R: React-based, S+R: Sense and React integrated

| Countermeasure | I | SI | NI | S | R | S+R |
|---|---|---|---|---|---|---|
| PUFMon [7] | | ✔ | ✔ | ✔ | | |
| Modulation detectors [8] | | ✔ | | ✔ | | |
| On-chip monitors (OCMs) [9] | | ✔ | ✔ | ✔ | | |
| Inductive impulse Self-destruction [10] | ✔ | ✔ | ✔ | | ✔ | |
| Memory zeroization [11] | ✔ | ✔ | ✔ | | ✔ | |
| NAND/NOR-based polymorphic latch [5] | | ✔ | ✔ | | | ✔ |
| Proposed polymorphic latch and registers | | ✔ | ✔ | | | ✔ |

### C. Polymorphic Gate Operation and Design

Polymorphic gates change their behavior based on changes in external factors. Recently, a novel approach based on multi-threshold null convention logic (MTNCL) was proposed, which leverages supply voltage sensitivity to control circuit functionality [12]. In this method, a control transistor pair determines whether the pull-up or pull-down network is activated, with the threshold voltage drop of a diode-connected NMOS transistor playing a crucial role in creating polymorphism in logic designs which we adopt. In our work, we apply genetic algorithm (GA) to fine-tune transistor sizes in polymorphic gates to achieve polymorphic behavior while minimizing transistor size. Key configuration parameters include selecting which transistors to evolve, population size, tournament size, mutation rate, and the range for transistor widths and lengths.

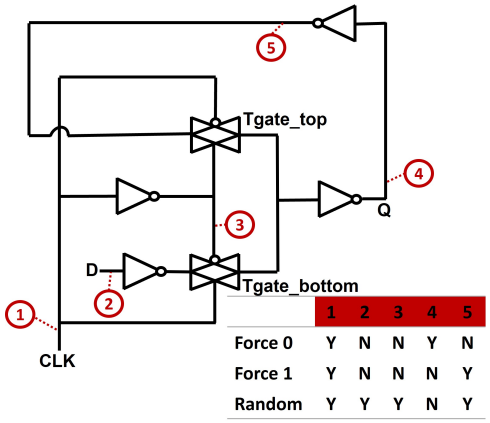
## III. Proposed Polymorphic Latches and Registers

The standard MUX-based latch shown in Fig. 2 can be turned into a polymorphic latch by placing one or more polymorphic buffer/always off (B/AO) gates [5] at strategic locations. Polymorphic B/AO gate functions as a buffer under normal voltage conditions, i.e., passes its input to its output. However, when the supply voltage is lowered below a threshold, the gate's output becomes always 0. The polymorphism threshold can be customized according to need by sizing transistors using genetic algorithm. Variants to customize state/output are detailed below:

**Force to '0' Polymorphic Latch Design:** The latch state can be forced to '0' by placing B/AO gates at locations 1 and 4 as shown in Fig. 2. B/A0 at 1 ensures only $Tgate_{top}$ is operational and B/A0 at 4 ensures output $Q$ is forced to '0' for the attack duration.

**Force to '1' Polymorphic Latch Design:** The latch state can be forced to '1' by placing B/AO gates at locations 1 and 5 as shown in Fig. 2. B/A0 at 5 ensures output $Q$ is forced to '1' for the attack duration.

**Randomize State Polymorphic Latch Design:** The latch state can be randomized by placing B/AO gates at locations $1, 2, 3$ and 5. B/AO at 1 and 3 ensures that both transmission gates are operational. B/AO




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Force 0 | Y | N | N | Y | N |
| Force 1 | Y | N | N | N | Y |
| Random | Y | Y | Y | N | Y |

Fig. 2: Standard MUX-based latch can be turned into polymorphic latch with customizable output by placing polymorphic B/AO gates at locations 1 to 5.

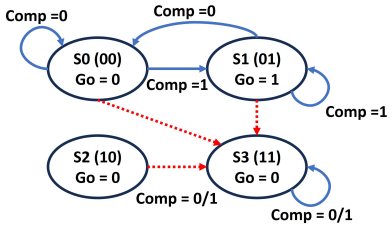



Fig. 3: Polymorphic finite state machine (FSM) for secure boot. The red dotted lines indicate state transitions to blackhole state $S3(11)$ from polymorphic behavior triggered by VFI attack.

at 2 ensures logic '1' is passed through $Tgate_{bottom}$ and B/AO at 5 ensures logic '0' is passed through $Tgate_{top}$ for the attack duration which leads to a race condition. $Q$ settles to random value depending on process variation.

**Extensions to Polymorphic Registers:** Polymorphic registers can be obtained by connecting any polymorphic latch in series with a standard latch. The two latches should have opposite clock signals.

## IV. Proposed Polymorphic FSMs

Polymorphic latches and registers can be used in two anti-VFI applications. First, they can be used to replace the standard registers that store security assets (e.g., keys, intermediate crypto applications, etc.). Second, they can replace state registers in FSMs and on-chip controllers. When the chip is undervolted during a VFI attack, the register values can be altered in randomized or controlled ways depending on the latch design chosen from Section III. The implications for the first and second applications are discussed in Section V and in the rest of this section, respectively.

### A. Polymorphic Control FSM for Secure Boot

A standard verification FSM for secure bootload and its polymorphic version is shown in Fig. 3. The FSM takes a comparator output, $Comp$ as input in $S0(00)$ state, where the comparator output 0 signifies password/signature mismatch and 1 signifies a match. If there is a match, FSM transitions from $S0(00)$ to $S1(01)$ otherwise it stays in $S0(00)$. When FSM in state $S1(01)$ and $Comp$ is still 1, it stays in $S1(01)$ and gives a signal $Go$ which initiates secure boot load. Here, attacker can force $Comp$ to 1 using VFI which initiates secure boot-load although authentication failed.

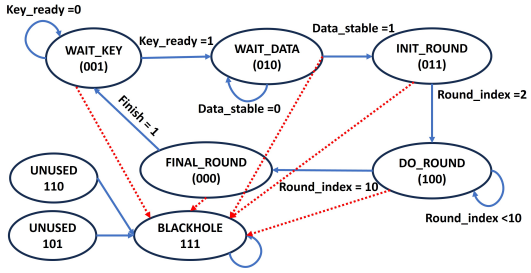



Fig. 4: Finite state machine (FSM) for AES controller with added states for blackhole state, 111. The red dotted lines indicate state transitions from polymorphic behavior that are triggered by VFI.

TABLE III: Power, performance, and area (PPA) comparison of registers taken at nominal conditions (1.1V, 27°C).

| Parameters | Standard MUX | Polymorphic MUX-based Registers Force to '0' | Force to '1' | Randomize |
|---|---|---|---|---|
| Area | 12 $\mu m^2$ | 26.4 $\mu m^2$ | 26.4 $\mu m^2$ | 43 $\mu m^2$ |
| $CLK2Q$ Delay | 28 ps / | 94 ps / | 93 ps / | 97 ps / |
| (Rise / Fall) | 36 ps | 99 ps | 103 ps | 128 ps |
| Setup Time | 26 ps | -41 ps | -40 ps | -41 ps |
| Hold Time | 4 ps | 72 ps | 70 ps | 71 ps |
| Average Power | 800 nW | 25 uW | 12.6 uW | 52.2 uW |

In the polymorphic version, $S3(11)$, a blackhole state is created by replacing the registers with polymorphic Force to '1' counterparts. Here any VFI attempt forces FSM to $S3(11)$ until system reset. Alternatively, the blackhole state could also be the reset state.

### B. Polymorphic Control FSM for Cryptographic Module

FSM controllers of cryptographic implementations such as AES, DES or RSA can also be protected. For example, Fig. 4 shows the control FSM for an AES implementation with polymorphic countermeasure. Here, in the original FSM, attacker can steal the key by performing VFI when the FSM transitions from 001 to 010. If a fault occurs after key is loaded in 001, to force FSM state to 000, attacker has access to the state register and it is possible to figure out the key using mathematical analysis.

Here, replacing the three registers with polymorphic Force to '1' variants will ensure that any VFI attempt results in the FSM to enter the blackhole state 111 until reset. *It is imperative to make sure that the FSM does not recover without reset so that attack cannot be carried out after recovery.*

## V. Results and Discussion

### A. Simulation Setup

Cadence Virtuoso is used for simulation purposes and all the designs were implemented in $45nm$ technology node using the $gpdk045$ library. Cadence Virtuoso ADE is used to measure the performance, power, and area (PPA) and delay characteristics of each latch and register. To assess the reliability of the latch and register, Monte Carlo analysis is conducted in Cadence Virtuoso's ADE XL simulation environment. Both process variation and transistor mismatch are accounted for in the Monte Carlo analysis. A temperature sweep is performed in Cadence Virtuoso ADE to analyze the resilience of the latches/registers to temperature fluctuations.

### B. MUX-based Polymorphic Register Simulation

*1) Power, Performance and Area (PPA) Overhead:* All simulations for the PPA overhead for MUX-based registers are conducted under

TABLE IV: Reliability analysis of registers under worst case normal operating condition ($V_{DD}$ lowered by $10\%$ at $84^0C$) and under attack condition ($V_{DD}$ lowered to $0.55V$) with Monte Carlo simulations.

| Design | No Attack | Under Attack | |
|---|---|---|---|
| | Retained | Flipped | Retained |
| Standard MUX-based | 100% | 0% | 100% |
| Force to '0' Polymorphic | 100% | 100% | 0% |
| Force to '1' Polymorphic | 100% | 100% | 0% |
| Randomize state Polymorphic | 100% | 48.5% | 51.5% |

TABLE V: Overhead in terms of area, power and corruption rate in terms of hamming distance (HD) achieved using proposed MUX-based registers in AES, DES and RSA crypto modules.

| Benchmark | Register type | Area Increase | Power Increase | Corruption Rate |
|---|---|---|---|---|
| AES 128 | Force to '0' | 6.24% | 3.7× | 50% |
| | Force to '1' | 6.24% | 2.3× | 50% |
| | Randomize | 13.5% | 6.8× | 49% |
| DES 64 | Force to '0' | 6.2% | 2.2× | 52% |
| | Force to '1' | 6.2% | 1.5× | 51% |
| | Randomize | 13.6% | 3.5× | 50% |
| RSA 512 | Force to '0' | 0.1% | 2.2× | 49% |
| | Force to '1' | 0.1% | 1.5× | 48% |
| | Randomize | 0.2% | 3.5× | 51% |

nominal conditions at $27^{\circ}$C and standard 1.1V operation. The PPA measurements for the standard and polymorphic MUX-based registers are demonstrated in Table III. The MUX-based registers are more efficient compared to the NAND/NOR-based counterparts [5] in terms of area and timing overhead.

*2) Impacts of Temperature:* The response time is measured as the time from the drop in supply voltage to the output being flipped. The Force to '0' and Force to '1' polymorphic registers show a worst-case response time of 1.2ns and 800ps respectively at $0^{\circ}$C while the room temperature response time is 1.1ns and 753ps respectively. The randomize state polymorphic register show a worst-case response time of 14.7ns at $84^{\circ}$C with the room temperature response time of 13.25ns. Even at the worst case, the registers are fast enough to destroy data to thwart VFI attacks.

*3) Reliability over PVT Variations:* All MUX-based polymorphic register designs including the standard design are able to latch and hold data for 100% of simulation points at normal operating voltage of 1V. This is expected for standard register, but also illustrates that the MUX-based polymorphic registers can be reliably deployed.

The second Monte Carlo simulation is run under simulated attack conditions. The results are shown in Table IV. The standard MUX-based register retains data 100% of the times, making it susceptible to attack. Force to '0' and force to '1' polymorphic registers show perfect reliability by forcing output to logic $low$ and $high$ respectively for all Monte Carlo simulation points. The randomize state polymorphic register randomizes the state 97% of the times. In short, all MUX-based polymorphic designs show improvement in all fields of area, power, performance and reliability over the NAND/NOR-based counterparts from [5].

### C. Simulation of Corruption in Cryptographic Benchmarks

VFI attacks are simulated against AES, DES, and RSA benchmarks containing polymorphic registers. The number of registers replaced by polymorphic registers in each are 128, 56 and 48, respectively.

The area, power increase, and corruption rate of resulting ciphertext are provided in Table V. The randomize state polymorphic register has three internal B/AO gates leading to significant power overhead. All MUX-based registers support gigahertz operation for their negative setup time. Overall area, timing and power overhead is improved compared to NAND/NOR polymorphic counterparts.

The corruption rate is measured as the percent change in the hamming distance (HD) between glitch-free ciphertext and glitch-induced ciphertext. Overall, the corruption rate hovers around the optimal 50% (48% to 52%), indicating maximum uncertainty in the stored data and making it effectively irrecoverable via VFI attack.

## VI. Conclusion

Non-invasive voltage fault injection attacks threaten the security of modern processors and existing solutions are not effective in thwarting such threat. In this work, the MUX-based polymorphic registers introduced are capable of simultaneously detecting and reacting to VFI attacks, destroying critical control operations and secret values. The effectiveness of such countermeasure is assessed in established cryptographic benchmarks. Further investigation of the performance of polymorphic registers with silicon results has the potential to pave the path to widespread adoption.

## VII. Acknowledgments

This effort was sponsored by NSF under grant #2150122. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

## References

[1] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "Voltjockey: Breaching trustzone by software-controlled voltage manipulation over multi-core frequencies," *Proceedings of the 2019 ACM SIGSAC Conference on CCS*, 2019.

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

[3] Z. Chen, G. Vasilakis, K. Murdock, E. Dean, D. F. Oswald, and F. D. Garcia, "Voltpillager: Hardware-based fault injection attacks against intel sgx enclaves using the svid voltage scaling interface," in *USENIX Security Symposium*, 2021.

[4] R. Buhren, H. N. Jacob, T. Krachenfels, and J.-P. Seifert, "One glitch to rule them all: Fault injection attacks against amd's secure encrypted virtualization," *Proceedings of the 2021 ACM SIGSAC Conference on CCS*, 2021.

[5] A. Cannon, T. Farheen, S. Roy, S. Tajik, and D. Forte, "Protection against physical attacks through self-destructive polymorphic latch," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.

[6] C. O'Flynn, "Fault injection using crowbars on embedded systems," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 810, 2016.

[7] S. Tajik, J. Fietkau, H. Lohrke, J.-P. Seifert, and C. Boit, "Pufmon: Security monitoring of fpgas using physically unclonable functions," *2017 IEEE 23rd International Symposium IOLTS*, pp. 186–191, 2017.

[8] S. Roy, T. Farheen, S. Tajik, and D. Forte, "Self-timed sensors for detecting static optical side channel attacks," in *2022 23rd ISQED*. IEEE, 2022, pp. 1–6.

[9] M. Nagata, T. Miki, and N. Miura, "On-chip physical attack protection circuits for hardware security," in *2019 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2019, pp. 1–6.

[10] S. Tada, Y. Yamashita, K. Matsuda, M. Nagata, K. Sakiyama, and N. Miura, "Design and concept proof of an inductive impulse self-destructor in sense-and-react countermeasure against physical attacks," *Japanese Journal of Applied Physics*, vol. 60, no. SB, p. SBBL01, 2021.

[11] A. Srivastava and P. Ghosh, "An efficient memory zeroization technique under side-channel attacks," in *2019 32nd International Conference on VLSID*. IEEE, 2019, pp. 76–81.

[12] C. Bernard, W. Bryant, R. Becker, and J. Di, "Design of asynchronous polymorphic logic gates for hardware security," in *2021 IEEE HPEC*. IEEE, 2021, pp. 1–5.