Karlsruhe Institute of Technology

Physical Security of Emerging AI Hardware Accelerators: From Vulnerability to Countermeasures

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)
genehmigte

Dissertation
von
Brojogopal Sapui
aus Kolkata, India

Tag der mündlichen Prüfung: 16. January 2026

1. Referent: Prof. Dr. Mehdi B. Tahoori
Karlsruhe Institute für Technology (KIT)

2. Referent: Prof. Dr. Amir Moradi
TU Darmstadt

Acknowledgement

Reflecting on this journey, I feel incredibly fortunate to have been surrounded by extraordinary individuals whose support, kindness, and patience made this dissertation possible.

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Dr. Mehdi Tahoori. I could not have asked for a better mentor throughout my PhD. His scientific insight, thoughtful feedback, and unwavering support-especially during the more difficult phases of this work-have shaped both this dissertation and my development as a researcher.

My sincere thanks go to all my colleagues at the Chair of Dependable Nano Computing (CDNC). I am especially grateful to our secretary, Ms. Iris Schröder-Piepka, for her tireless efforts in keeping everything running smoothly in the background. I would like to thank Dr.-Ing. Jonas Krautter and Dr.-Ing. Christopher Münch for their generous help, insightful discussions, and willingness to answer my questions, even while they were already transitioning to positions outside KIT.

Most importantly, I would like to express my heartfelt gratitude to my family. I am profoundly thankful to my parents, whose unconditional love, sacrifices, and belief in me have been a constant source of strength. I am equally grateful to my elder sister and my dear nieces, whose affection and encouragement brought warmth and joy across the distance. I also owe special thanks to my dear friends Priyanjana Pal and Surendra Hemaram, who were like family to me during this journey and whose friendship, conversations, and humour helped me through many difficult days. I am likewise grateful for the friendship and support of my wonderful colleagues and friends already mentioned above-including Atousa Jafari, Vincent Meyers, Sina Bakhtavari Mamaghani, Seyedehmaryam Ghasemi, Shanmukha Mangadahalli Siddaramu, Ali Nezhadi Khelejani, Gürol Saglam, Maha Shatta, Zhe Zhang, Simon Pankner, Tara Gheshlaghi, Aradhana Dube, Mahboobe Sadeghipourrudsari and Mahta Mayahinia-who each, in their own way, made this path lighter and more enjoyable.

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben haben und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen – die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Karlsruhe, 21. November 2025
Brojogopal sapui

Abstract

The widespread deployment of Artificial Intelligence (AI) at the edge has driven a paradigm shift toward domain-specific hardware accelerators. Emerging architectures-ranging from Analog Compute-in-Memory (CiM) and Non-Volatile Memory (NVM)-centric processors to Neuromorphic Computing and Hyperdimensional Computing (HDC)-promise orders-of-magnitude improvements in energy efficiency and latency. However, while these substrates are often used for their intrinsic robustness to noise and stochastic process variations, this dissertation demonstrates that such robustness does *not* translate into resilience against targeted physical attacks. This work systematically explores the physical security gap in post-Von Neumann computing, establishing that the very physical properties enabling efficiency often introduce novel, exploitable attack surfaces.

To address these challenges, this thesis develops a set of cross-layer analysis frameworks and countermeasures, spanning device physics, circuit simulation, and real-hardware validation. First, the research investigates Analog Compute-in-Memory (CiM) based on ReRAM and STT-MRAM. By developing a physics-aware simulation framework, it is shown that analog non-idealities-such as device-to-device variability and random telegraph noise-manifest as data-dependent leakage signatures. These signatures allow adversaries to recover fixed weights using correlation-based and profiling attacks, enabling the extraction of proprietary neural network models.

Second, the thesis uncovers a critical vulnerability in processor-centric NVM architectures. Through real-hardware experiments on STT-MRAM, a novel class of persistent fault attacks is demonstrated. By targeting the specific magnetic commit window of MRAM writes with nanosecond-scale voltage glitches, an attacker can induce stable, non-volatile bit corruptions in stored cryptographic keys. This persistence mechanism collapses the data complexity required for differential fault analysis (DFA) by orders of magnitude, enabling full AES key recovery with fewer than 20 ciphertext pairs.

Third, the security of Neuromorphic Computing is analyzed through the introduction of *FlexSpy*, the first design-time security framework for thin-film transistor (TFT)-based Spiking Neural Networks (SNNs). The analysis reveals a unique *quasi-DC* leakage mechanism inherent to event-driven processing on flexible substrates. Despite the absence of internal triggers, an attacker can exploit this leakage to infer input classes and recover layer-wise spiking activity with high fidelity. Lightweight circuit-level countermeasures are proposed that suppress this leakage by up to 70% with minimal power overhead.

Finally, the thesis presents a comprehensive security evaluation of FPGA-based Hyperdimensional Computing (HDC). It exposes a “robustness mismatch,” where HDC’s algorithmic error tolerance masks severe physical vulnerabilities. The work demonstrates successful Intellectual Property (IP) theft via deep-learning-assisted side-channel analysis and remote, internal sensing attacks using on-chip time-to-digital converters (TDCs). Furthermore, a targeted fault injection methodology (*HyFault*) is developed, capable of inducing precise misclassifications. In response, effective defenses-including dynamic masking and hypervector randomization-are introduced to restore security.

Collectively, these contributions provide a foundational understanding of the physical security risks in emerging AI hardware, offering actionable design guidelines to ensure that the next generation of efficient edge accelerators is secure by design.

Zusammenfassung

Die weit verbreitete Nutzung von Künstlicher Intelligenz (KI) am Randbereich (*Edge*) hat einen Paradigmenwechsel hin zu domänenspezifischen Hardware-Beschleunigern ausgelöst. Neuartige Architekturen - von Analog *Compute-in-Memory* (CiM) und nichtflüchtigkeitsspeicherzentrierten (*Non-Volatile Memory*, NVM) Prozessoren bis hin zu Neuromorpher Datenverarbeitung (NC) und Hyperdimensional Computing (HDC) - versprechen Größenordnungen an Verbesserungen hinsichtlich Energieeffizienz und Latenz. Obwohl diese Substrate häufig aufgrund ihrer intrinsischen Robustheit gegenüber Rauschen und stochastischen Prozessvariationen eingesetzt werden, zeigt diese Dissertation, dass eine solche Robustheit *keinesfalls* zu einer Widerstandsfähigkeit gegenüber gezielten physikalischen Angriffen führt. Die Arbeit untersucht systematisch die Sicherheitslücke in post-von-Neumann-Systemen und zeigt, dass gerade die physikalischen Eigenschaften, die Effizienz ermöglichen, neuartige und ausnutzbare Angriffsflächen eröffnen.

Um diese Herausforderungen anzugehen, entwickelt diese Dissertation eine Reihe von schichtübergreifenden Analyseframeworks und Gegenmaßnahmen, die von der Bauelementphysik über die Schaltungssimulation bis hin zur Validierung auf realer Hardware reichen. Zunächst untersucht die Arbeit Analog-*Compute-in-Memory* (CiM) auf Basis von ReRAM und STT-MRAM. Durch die Entwicklung eines physikbewussten Simulationsframeworks wird gezeigt, dass analoge Nichtidealitäten - wie Bauelement-zu-Bauelement-Variabilität und *Random Telegraph Noise* - datenabhängige Leckagesignaturen erzeugen. Diese Signaturen ermöglichen es Angreifern, stationäre Gewichte mittels korrelationsbasierter und profilierender Seitenkanalangriffe zurückzugewinnen und somit proprietäre neuronale Netzwerkmodelle auszulesen.

Zweitens deckt die Dissertation eine kritische Schwachstelle in prozessorzentrierten NVM-Architekturen auf. Durch Realhardware-Experimente an STT-MRAM wird eine neuartige Klasse von *persistenten Fehlerangriffen* gezeigt. Durch das präzise Anvisieren des magnetischen Commit-Fensters des MRAM-Schreibvorgangs mit Spannungsglitches im Nanosekundenbereich kann ein Angreifer stabile, nichtflüchtige Bitfehler in gespeicherten kryptografischen Schlüsseln erzeugen. Dieser Persistenzmechanismus reduziert die Datenkomplexität für differentielle Fehleranalyse (DFA) um mehrere Größenordnungen und ermöglicht eine vollständige AES-Schlüsselrekonstruktion mit weniger als 20 Chiffretextpaaren.

Drittens wird die Sicherheit der Neuromorphen Datenverarbeitung anhand der Einführung von *FlexSpy* analysiert, dem ersten Entwurfszeit-Sicherheitsframework für dünnschichttransistorbasierte (TFT) Spiking Neural Networks (SNNs). Die Analyse zeigt einen einzigartigen *quasi-DC*-Leckagemechanismus, der inhärent mit ereignisgetriebener Verarbeitung auf flexiblen Substraten verbunden ist. Trotz des Fehlens interner Trigger kann ein Angreifer diese Leckage ausnutzen, um Eingangsklassen abzuleiten und schichtweise Spiking-Aktivität mit hoher Genauigkeit zu rekonstruieren. Leichte Schaltungs-Gegenmaßnahmen werden vorgeschlagen, die diese Leckage um bis zu 70% reduzieren - bei minimalem zusätzlichen Energieaufwand.

Abschließend präsentiert die Dissertation eine umfassende Sicherheitsanalyse hardwarebeschleunigter Hyperdimensional Computing (HDC) Implementierungen auf FPGAs. Die Arbeit zeigt eine "*Robustheitsdiskrepanz*", bei der die algorithmische Fehlertoleranz von HDC schwerwiegende physikalische Verwundbarkeiten verdeckt. Die Ergebnisse demonstrieren erfolgreichen Diebstahl geistigen Eigentums durch Deep-Learning-unterstützte Seitenkanalanalyse sowie durch entfernte, interne Messangriffe mittels *On-Chip Time-to-Digital Converters* (TDCs). Darüber hinaus wird eine gezielte Fehlerinjektionsmethodik (*HyFault*) entwickelt, die präzise Fehlklassifikationen hervorrufen kann. Als Reaktion werden wirksame Gegenmaßnahmen - einschließlich dynamischem Masking und Hypervektor-Randomisierung - vorgestellt, die die Sicherheit maßgeblich wiederherstellen.

Insgesamt liefern diese Beiträge ein grundlegendes Verständnis der physikalischen Sicherheitsrisiken in aufkommenden KI-Hardwarebeschleunigern und bieten konkrete Entwurfsrichtlinien, um sicherzustellen, dass die nächste Generation energieeffizienter Edge-Beschleuniger *von Grund auf sicher* gestaltet wird.

Contents

List of Figures

List of Tables

Acronyms

a-IGZO	amorphous Indium-Gallium-Zinc Oxide.
ACIM	Analog Compute-in-Memory.
ADC	Analog-to-Digital Converter.
AES	Advanced Encryption Standard.
ALU	Arithmetic Logic Unit.
ANN	Artificial Neural Network.
ASIC	Application-Specific Integrated Circuit.
AXI	Advanced eXtensible Interface.
BER	Bit Error Rate.
BL	Bitline.
BRAM	Block RAM.
CAM	Content Addressable Memory.
CiM	Compute-in-Memory.
CMOS	Complementary Metal–Oxide–Semiconductor.
CNN	Convolutional Neural Network.
CPA	Correlation Power Analysis.
CPU	Central Processing Unit.
CRC	Cyclic Redundancy Check.
DAC	Digital-to-Analog Converter.
DC	Direct Current.
DFA	Differential Fault Analysis.
DL-SCA	Deep-Learning-based Side-Channel Analysis.
DMA	Direct Memory Access.
DNN	Deep Neural Network.
DPA	Differential Power Analysis.
DRAM	Dynamic Random-Access Memory.
DSA	Domain-Specific Architecture.

ECC	Error-Correcting Code.
EM	Electromagnetic.
FE	Flexible Electronics.
FI	Fault Injection.
FIB	Focused Ion Beam.
FPGA	Field-Programmable Gate Array.
HD	Hamming Distance.
HDC	Hyperdimensional Computing.
HW	Hamming Weight.
ILA	Integrated Logic Analyzer.
ISI	Inter-Spike Interval.
KCL	Kirchhoff's Current Law.
LDA	Linear Discriminant Analysis.
LIF	Leaky Integrate-and-Fire.
LLM	Large Language Model.
MAC	Multiply–Accumulate.
MI	Mutual Information.
MIM	Metal–Insulator–Metal.
ML	Machine Learning.
MLP	Multi-Layer Perceptron.
MN	Measurement Noise.
MRAM	Magnetoresistive Random-Access Memory.
MTJ	Magnetic Tunnel Junction.
MVM	Matrix-Vector Multiplication.
NC	Neuromorphic Computing.
NoC	Network-on-Chip.
NVM	Non-Volatile Memory.
PCA	Principal Component Analysis.
PCB	Printed Circuit Board.
PCM	Phase-Change Memory.
PDK	Process Design Kit.

PDN	Power Delivery Network.
PFA	Persistent Fault Analysis.
PMOD	Peripheral Module Interface.
PUF	Physical Unclonable Function.
PVT	Process, Voltage, and Temperature.
QDA	Quadratic Discriminant Analysis.
RC	Resistance–Capacitance.
ReRAM	Resistive Random-Access Memory.
RNN	Recurrent Neural Network.
RTN	Random Telegraph Noise.
SCA	Side-Channel Analysis.
SLI	Spike-Leakage Index.
SNM	Static Noise Margin.
SNN	Spiking Neural Network.
SNR	Signal-to-Noise Ratio.
SoC	System-on-Chip.
SPA	Simple Power Analysis.
SPFA	Statistical Persistent Fault Analysis.
SPI	Serial Peripheral Interface.
STT-MRAM	Spin-Transfer-Torque Magnetoresistive RAM.
TDC	Time-to-Digital Converter.
TFT	Thin-Film Transistor.
TIA	Transimpedance Amplifier.
TMR	Tunnel Magnetoresistance.
TVLA	Test Vector Leakage Assessment.
UART	Universal Asynchronous Receiver-Transmitter.
VDD	Supply Voltage.
WL	Wordline.

Part I.

Preliminaries

1. Introduction and Motivation

The trajectory of modern computing is currently navigating a critical inflection point, driven by the convergence of physical scaling limits and the explosive computational demands of Artificial Intelligence (AI). For over half a century, the information technology sector has relied upon the consistency of Moore's Law and Dennard scaling to deliver exponential improvements in performance and energy efficiency. However, as transistor feature sizes approach atomic scales, these reliable scaling laws have faltered, giving rise to the "Dark Silicon" era where thermal constraints prevent the simultaneous utilization of all on-chip resources. Concurrently, the nature of computational workloads has fundamentally shifted. The dominance of control-heavy, sequential logic has transferred ground to data-intensive, massively parallel workloads characteristic of AI and other machine learning paradigms. This mismatch between traditional architectural principles and modern workload requirements has precipitated a crisis centered on the movement of data, famously termed the "Memory Wall" [2].

This dissertation addresses the urgent need to re-evaluate hardware security in light of the architectural solutions emerging to solve the Memory Wall crisis. As the industry pivots toward non-Von Neumann architectures, specifically CiM Analog Compute-in-Memory (CiM), Neuromorphic Computing, and Hyperdimensional Computing (HDC), it undoubtedly invalidates the abstraction layers that have historically underpinned hardware security models [44], [48], [63] [21], [88]. By transitioning from digital, deterministic logic to analog, stochastic, and approximate computing, these emerging accelerators expose novel physical attack surfaces that are poorly understood and inadequately protected. The following sections describe the context of this architectural shift, define the specific security gaps this research addresses, and outline the technical contributions that form the core of this thesis.

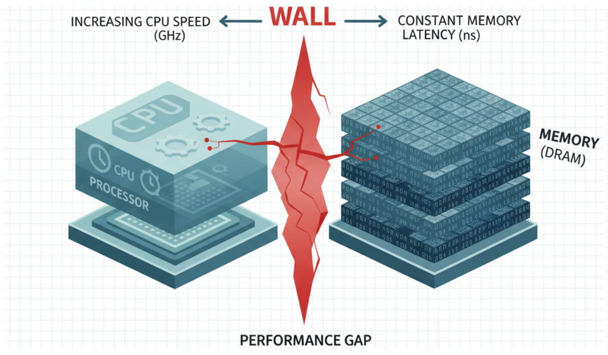
1.1. Motivation

1.1.1. The Crisis: The Memory Wall and the Energy Efficiency Gap

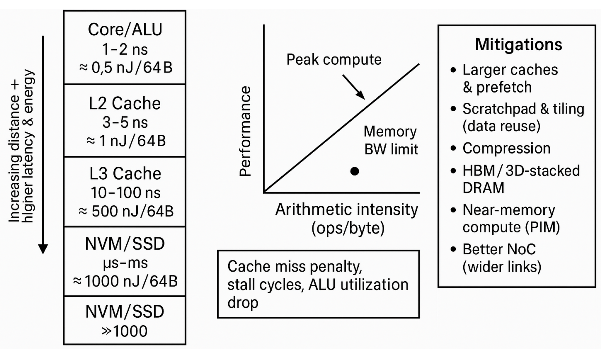
The Von Neumann architecture, characterized by the physical and logical separation of the Central Processing Unit (CPU) and the memory hierarchy, has served as the pillar of general-purpose computing for decades. This separation necessitates the continuous transfer of instructions and data between storage units (DRAM) and processing units via a shared bus. While effective for conventional workloads where data reuse is low and control logic is complex, this architecture creates a "data traffic jam" for AI applications.

Deep Learning models, particularly modern Transformers and Convolutional Neural Networks (Convolutional Neural Networks (CNNs)), are characterized by billions of parameters that must be fetched from memory to perform Matrix-Vector Multiplications (Matrix-Vector Multiplications (MVMs)). The energy cost of this data movement has now eclipsed the cost of the computation itself. Empirical analyses indicate that fetching a single operand from off-chip DRAM can consume between 100 to 500 times more energy than performing the floating-point operation on that data [36]. Consequently, in data-centric AI workloads, the processor spends a significant portion of its time idle, waiting for data to arrive, while the system's power budget is dominated by the interconnects rather than the arithmetic logic units (Arithmetic Logic Units (ALUs)).

This phenomenon, known as the Memory Wall, creates a formidable barrier to deploying advanced AI at the edge. Edge devices, ranging from autonomous vehicles and industrial IoT sensors to wearable health monitors, operate under stringent power envelopes and latency constraints that preclude the use of power-hungry, bandwidth-limited Von Neumann architectures. To bridge this gap, the semiconductor



(a) CPU–memory performance gap (“memory wall”).


Increasing distance + higher latency & energy
Core/ALU 1-2 ns ≈ 0,5 nJ/64 B
L2 Cache 3-5 ns ≈ 1 nJ/64 B
L3 Cache 10-100 ns ≈ 500 nJ/64 B
NVM/SSD μs-ms ≈ 1000 nJ/64 B
NVM/SSD ≫1000
Performance
Peak compute
Memory BW limit
Arithmetic intensity (ops/byte)
Cache miss penalty, stall cycles, ALU utilization drop
Mitigations
• Larger caches & prefetch
• Scratchpad & tiling (data reuse)
• Compression
• HBM/3D-stacked DRAM
• Near-memory compute (PIM)
• Better NoC (wider links)

(b) Latency/energy hierarchy and roofline limit with mitigation levers.

Figure 1.1.: Visualization of the memory wall: CPU performance increases rapidly while DRAM latency stays nearly constant, creating a fundamental bottleneck. The accompanying memory hierarchy and roofline model highlight latency/energy scaling across cache levels and motivate mitigation techniques such as data reuse, compression, near-memory compute, and wider NoC.

industry is aggressively pursuing Domain-Specific Architectures (Domain-Specific Architectures (DSAs)) that minimize data movement by physically co-locating memory and computation.

1.1.2. The Paradigm Shift: Emerging Hardware Accelerators

This thesis specifically investigates three classes of emerging accelerators that represent the most promising solutions to the Memory Wall: Analog Compute-in-Memory (CiM), Neuromorphic Computing (Neuromorphic Computing (NC)), and Hyperdimensional Computing (HDC). Each of these paradigms leverages novel device physics or computational models to achieve orders-of-magnitude improvements in energy efficiency, yet each introduces distinct physical security vulnerabilities.

1.1.2.1. Analog Compute-in-Memory

Compute-in-Memory (CiM) fundamentally restructures the computing hierarchy by performing arithmetic operations directly within the memory array, thereby eliminating the need to shuttle weight data across the bus. While digital CiM implementations exist, Analog CiM is particularly compelling for AI inference due to its ability to perform Matrix-Vector Multiplication (MVM) in the analog domain with extreme density. In an Analog CiM crossbar utilizing Non-Volatile Memory (NVM) technologies, such as Resistive RAM (ReRAM) or Spin-Transfer Torque Magnetic RAM (STT-MRAM), synaptic weights are encoded as the physical conductance states of the memory cells [35], [41]. Input vectors are applied as voltages along the wordlines. Following Ohm’s Law and Kirchhoff’s Current Law, the accumulation of current along the bitlines naturally computes the dot product, the fundamental atomic operation of neural networks. This in-situ computation allows for massive parallelism and near-zero data movement energy. However, reliance on analog properties introduces device non-idealities, such as conductance variability, Random Telegraph Noise (Random Telegraph Noise (RTN)), and thermal sensitivity, that have profound implications for information leakage.

1.1.2.2. Neuromorphic Computing

Parallel to the architectural shift in high-performance computing is the material revolution enabling "ubiquitous intelligence" via Flexible Electronics (Flexible Electronics (FE)). Fabricated on substrates like polyimide using low-temperature processes, these devices utilize materials such as amorphous Indium-Gallium-Zinc Oxide (amorphous Indium-Gallium-Zinc Oxide (a-IGZO)) Thin-Film Transistors (Thin-Film Transistors (TFTs)). a-IGZO offers sufficiently high mobility for logic circuits while enabling mechanical flexibility, transparency, and low manufacturing costs. When combined with the computational paradigm of Spiking Neural Networks (Spiking Neural Networks (SNNs)), which process information as sparse, asynchronous events (spikes) rather than continuous values, flexible electronics promise to

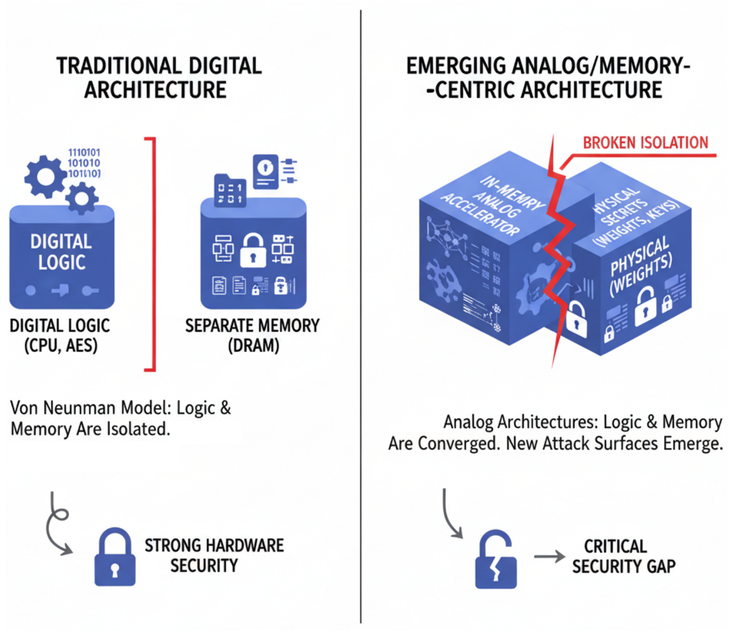

TRADITIONAL DIGITAL ARCHITECTURE
DIGITAL LOGIC
DIGITAL LOGIC (CPU, AES)
SEPARATE MEMORY (DRAM)
Von Neunman Model: Logic & Memory Are Isolated.
STRONG HARDWARE SECURITY
EMERGING ANALOG/MEMORY--CENTRIC ARCHITECTURE
BROKEN ISOLATION
IN-MEMORY ANALOG ACCELERATOR
PHYSICAL SECRETS (WEIGHTS, KEYS)
PHYSICAL (WEIGHTS)
Analog Architectures: Logic & Memory Are Converged. New Attack Surfaces Emerge.
CRITICAL SECURITY GAP

Figure 1.2.: Traditional digital architectures maintain strict physical separation between logic (e.g., CPUs, AES engines) and memory (e.g., DRAM), providing strong isolation and mature hardware-security guarantees. Emerging analog and memory-centric accelerators collapse this boundary by co-locating computation and storage. While this improves efficiency, it breaks architectural isolation and introduces new attack surfaces that expose physical weights, internal states, and stored secrets to leakage and fault-injection vulnerabilities.

bring efficient AI to conformal surfaces like smart skin, medical implants, and smart packaging. Unlike conventional CMOS, however, these flexible substrates lack robust packaging and sophisticated Power Delivery Networks (PDNs), creating a physical environment that is uniquely exposed to side-channel observation and manipulation [6], [23], [26].

1.1.2.3. Hyperdimensional Computing (HDC)

Hyperdimensional Computing (HDC) represents a radical departure from standard numerical representation. Inspired by the distributed and holographic nature of biological memory, HDC encodes information into high-dimensional pseudo-random vectors (hypervectors). Learning and inference are performed using computationally efficient bitwise operations (Binding via XOR, Bundling via Addition, and Similarity via Hamming Distance) rather than complex floating-point arithmetic. HDC is celebrated for its intrinsic algorithmic robustness; the distributed representation ensures that the failure of individual bits or moderate levels of noise does not catastrophically affect the system's accuracy. This makes HDC an ideal candidate for implementation on error-prone, low-power hardware such as emerging NVMs or FPGAs (FPGAs). However, as this thesis will demonstrate, this algorithmic error tolerance creates a "robustness Mismatch," potentially masking targeted attacks that exploit the specific hardware realization of HDC operations.

1.1.3. The Security Gap: Physical Vulnerabilities in the Post-Digital Era

While the performance and efficiency benefits of these emerging accelerators are extensively documented in literature, their physical security posture remains critically underexplored. The transition from digital, logic-centric Von Neumann systems to analog, memory-centric, and stochastic architectures creates a security gap. Traditional hardware security models, developed primarily for digital cryptographic circuits (e.g., AES on CMOS), rely on assumptions that are invalid in these new regimes.

Specifically, the physical proximity of memory and computation destroys the isolation that previously protected secrets (weights, keys) from computation-dependent leakage. In Analog CiM, the "leakage" is indistinguishable from the computation itself, as the supply current is a direct analog function of the sensitive weights. In NVM-based systems, the non-volatility of memory cells introduces the threat of "persistence," where a transient fault injection becomes a permanent data corruption, fundamentally altering the threat model for fault analysis. Furthermore, the deployment of AI at the edge places these devices in physically accessible environments, exposing them to sophisticated SCA) and (Fault Injection (FI) attacks that can extract proprietary models or corrupt inference integrity.

This dissertation is motivated by the premise that the true potential of emerging AI accelerators cannot be realized without a rigorous understanding of their physical vulnerabilities. We argue that security must be treated as a primary design constraint, co-optimized alongside power, performance, and area through a cross-layer approach that spans device physics to system architecture.

1.2. Problem Statement

The overarching problem addressed in this thesis is the inadequacy of current hardware security frameworks to protect emerging AI accelerators against physical attacks. The unique physical properties of NVM devices, the analog nature of CiM, the sparsity of SNNs, and the distributed representation of HDC introduce novel attack vectors that bypass standard digital countermeasures. This thesis decomposes this problem into four specific technical challenges.

1.2.1. The Failure of Digital Leakage Models in Analog Computing

Hardware security has historically relied on digital leakage models, such as the Hamming Weight (HW) or HD models, to quantify the correlation between power consumption and processed data. These models assume that power consumption is dominated by the charging and discharging of load capacitances during discrete digital switching events. However, in Analog CiM architectures utilizing ReRAM or STT-MRAM, computation is performed in the charge or current domain. The power consumption is determined by the summation of currents through a resistive crossbar array, governed by Ohm's Law and Kirchhoff's laws. This current is a continuous analog function of the input voltages and the programmed conductance states (weights). Furthermore, this signal is modulated by device-specific analog non-idealities:

Resistive RAM (ReRAM): Exhibits cycle-to-cycle variability and RTN due to the stochastic nature of conductive filament formation and rupture.

STT-MRAM: Subject to thermal fluctuations affecting the Tunnel Magnetoresistance (Tunnel Magnetoresistance (TMR)) ratio and stochastic switching probabilities.

Standard digital SCA methodologies fail to capture these complex, continuous-time interactions. There is a critical lack of physics-aware leakage models that can accurately predict the information leakage of Analog CiM macros or estimate the "Trace-to-Disclosure" metric for neural network weights stored in NVM arrays. Without such models, designers cannot effectively evaluate the security of CiM accelerators during the design phase.

1.2.2. The Threat of Persistence: Non-Volatility as an Attack Vector

The integration of NVM into processor-centric architectures, such as MRAM-based caches or embedded MRAM (eMRAM) for key storage, introduces a novel and potent attack surface: persistent faults. In traditional SRAM-based systems, a fault injected via voltage glitching or laser pulses is transient; it exists only until the data is overwritten or the device is reset. This requires attackers to precisely synchronize their injection for every single encryption or inference run they wish to corrupt. NVM technologies fundamentally alter this dynamic. A fault injected during the write operation of an STT-MRAM cell physically alters the magnetic orientation of the Magnetic Tunnel Junction (MTJ), permanently flipping the bit. This "persistent" fault remains active across power cycles and resets until it is explicitly rewritten. This persistence allows an attacker to inject a fault once during a "commit window and collect an arbitrary number of faulty outputs (ciphertexts or inference results) derived from the same static fault. It drastically reduce the data complexity required for differential analysis.

Existing security evaluations often overlook the specific physics of the STT-MRAM write operation, particularly the asymmetry between Parallel-to-Antiparallel and Antiparallel-to-Parallel switching, and the "commit window" vulnerability where the MTJ is most susceptible to voltage disturbances. Conventional countermeasures like redundancy (e.g., re-computing the result) are ineffective against persistent faults because the re-computation will use the same corrupted data, yielding the same valid (but incorrect) result.

1.2.3. Security Blind-Spots in Emerging Neuromorphic Hardware

The drive for ubiquitous AI has led to the development of emerging neuromorphic hardware based on thin-film transistors (TFTs). While a-IGZO TFTs enable low-cost, flexible and lightweight implementations, they operate under fundamentally different physical constraints than bulk silicon CMOS. Flexible substrates typically use simplified Power Delivery Networks (PDNs) with shared supply rails and lack rigid packaging and groundplanes that provide electromagnetic shielding in standard chips. When these substrates run event-driven Spiking Neural Networks (SNNs), a unique leakage mechanism emerges. SNNs process information via discrete spikes, meaning that the dynamic power consumption is directly proportional to the instantaneous firing rate of the network. In flexible electronics, the slower switching speeds and specific device physics of TFTs can manifest as "quasi-DC" leakage signatures that are highly correlated with the input stimuli and the network's internal state. Currently, there is no unified framework to model these substrate-specific leakages or to simulate the vulnerability of flexible SNNs prior to fabrication.

1.2.4. The "Robustness Mismatch" in Hyperdimensional Computing

FPGA-based accelerators for HDC are gaining traction for edge learning due to their efficiency. A common misconception in the HDC community is that the paradigm's inherent algorithmic robustness, its ability to tolerate Bit Error Rates (BERs), translates to hardware security. This thesis identifies this as the "Robustness Mismatch". While HDC is indeed resilient to random, uniformly distributed noise, it is fragile to targeted perturbations. The massive parallelism required by HDC (e.g., high-dimensional XOR arrays for binding operations) generates distinct, high-magnitude power consumption patterns. These patterns can be exploited by advanced Deep-Learning-based Side-Channel Analysis (DL-SCA) to recover the bits of secret hypervectors representing the learned model. Furthermore, the algorithmic tolerance to errors can be weaponized by an attacker to inject stealthy faults that force specific misclassifications without causing a noticeable degradation in overall system accuracy or triggering crash-detection mechanisms. Quantifying the limits of this robustness and identifying the precise hardware structures (e.g., Popcount units) that act as a leakage pipe is an open problem.

1.3. Research Objectives and Contributions

The primary objective of this dissertation is to establish a comprehensive, physically grounded security framework for emerging AI hardware accelerators. By bridging the gap between device physics, circuit design, and system architecture, this research aims to characterize the unique vulnerabilities of post-Von Neumann systems and develop cross-layer countermeasures.

Figure 1.3 maps the prior landscape against the central security gaps that this dissertation addresses. Gray dashed ovals represent existing work, whereas green ovals highlight the specific contributions and thesis results that fill these gaps. This visual summary illustrates how the present work spans Analog CiM, persistent NVM-based attacks, flexible neuromorphic circuits, and FPGA-based HDC accelerators.

The research is structured around four central Research Questions (RQs), each addressing a specific accelerator paradigm and security challenge.

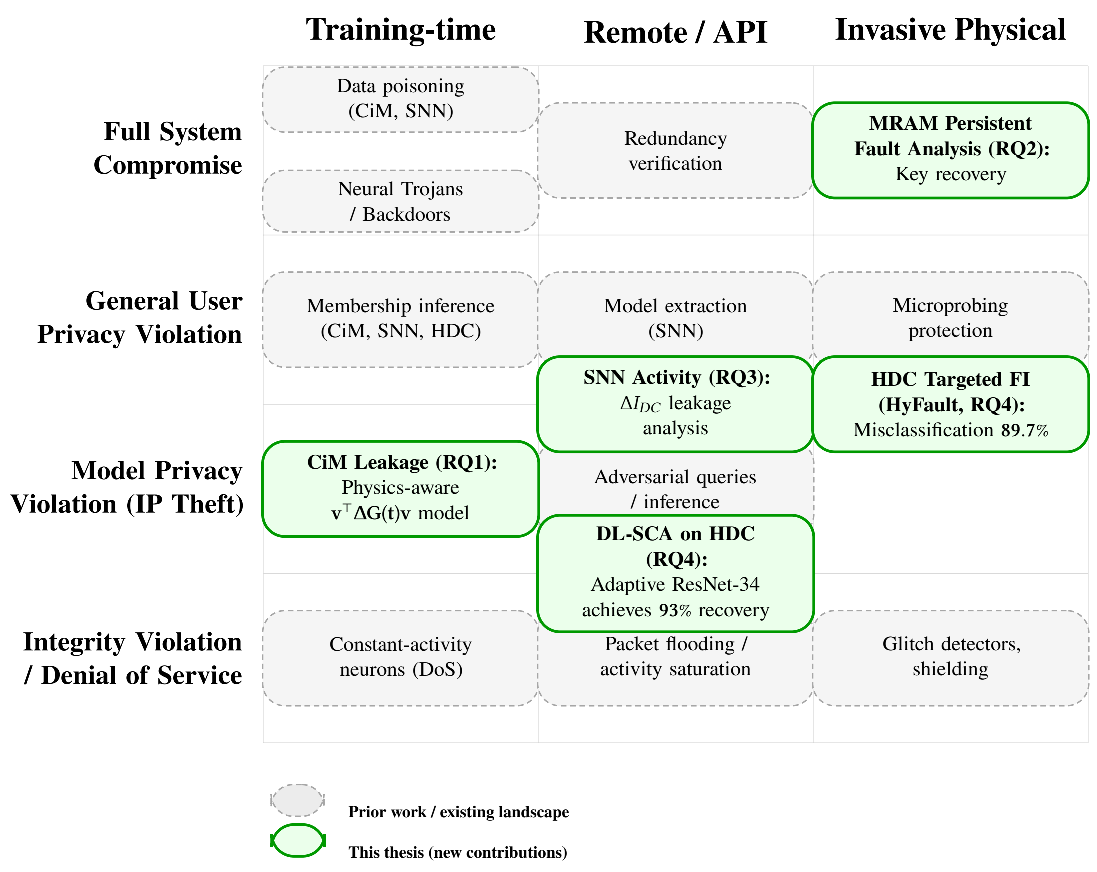

Training-time
Remote / API
Invasive Physical
Full System Compromise
General User Privacy Violation
Model Privacy Violation (IP Theft)
Integrity Violation / Denial of Service
Data poisoning (CiM, SNN)
Neural Trojans / Backdoors
Redundancy verification
MRAM Persistent Fault Analysis (RQ2): Key recovery
Membership inference (CiM, SNN, HDC)
Model extraction (SNN)
Microprobing protection
SNN Activity (RQ3): ΔI_{DC} leakage analysis
HDC Targeted FI (HyFault, RQ4): Misclassification 89.7%
CiM Leakage (RQ1): Physics-aware $v^\top \Delta G(t) v$ model
Adversarial queries / inference
DL-SCA on HDC (RQ4): Adaptive ResNet-34 achieves 93% recovery
Constant-activity neurons (DoS)
Packet flooding / activity saturation
Glitch detectors, shielding
Prior work / existing landscape
This thesis (new contributions)

Figure 1.3.: Coverage of research gaps by this thesis. The diagram maps the threat landscape across attack vectors (columns) and impact severity (rows). Green ovals denote the specific contributions of this work, filling critical gaps in prior research (gray dashed ovals).

1.3.1. RQ1: Vulnerabilities of Emerging NVM-based In-memory Computing (Analog)

Research Question: How do the fundamental device physics and analog non-idealities of ReRAM and STT-MRAM translate into exploitable side-channel leakage in Analog CiM architectures, and how can this leakage be modeled and mitigated?

Contribution (Chapter 3):

This chapter presents a comprehensive simulation-based analysis of side-channel leakage in Analog CiM accelerators.

Physics-Aware Modeling: We develop a hierarchical leakage model that propagates device-level non-idealities (e.g., ReRAM conductance fluctuations, MRAM thermal noise) up to the array and system level. This model explicitly captures the sensing current-leakage relationship and its modulation by device stochasticity.

Primitive-Level Analysis: We analyze the vulnerability of specific analog compute primitives, including Scouting Logic (XOR), MAC operations with non-linear activation functions (tanh), and Content Addressable Memory (CAM) based HDC.

Quantification: We introduce metrics to quantify the "Trace-to-Disclosure" for weight recovery, demonstrating that analog non-idealities, while typically considered noise, can actually create distinct leakage fingerprints that facilitate model extraction.

Decision Failure Analysis: We investigate the reliability-security interface, showing how "decision failure", incorrect sensing due to process variation, can be correlated with input patterns to reveal sensitive information.

1.3.2. RQ2: Persistent Faults in Processor-Centric NVM Architectures (MRAM)

Research Question: Can the unique write-physics of STT-MRAM, specifically the commit window and switching asymmetry, be exploited to induce persistent faults, and how does this persistence alter the landscape of Persistent Fault Analysis (PFA), generally discussed in digital leakage characterization?

Contribution (Chapter 4):

This chapter moves from simulation to real-hardware validation, focusing on STT-MRAM in processor-centric systems.

Commit Window Characterization: We experimentally characterize the "commit window" of STT-MRAM chips, the specific temporal window during a write cycle where the MTJ is most susceptible to voltage glitches. We demonstrate the asymmetry in fault injection probability for magnetic polarity transitions.

Persistent Fault Analysis (PFA): Using an FPGA-based platform with an MRAM-backed AES implementation, we demonstrate a practical PFA attack [69], [79]. We show that a single successful glitch during the key loading phase induces a persistent fault that enables full key recovery with a minimal number of ciphertext pairs (12-17 pairs), significantly lower than the thousands required for standard DFA.

Countermeasures: We propose architectural defenses, such as randomized commit timing (dithering the write pulse) and write-verification protocols, to mitigate the threat of persistent faults without abandoning the benefits of NVM.

1.3.3. RQ3: Security of Emerging Spiking Neuromorphic Computing (Flexible Edge-AI)

Research Question: What are the dominant leakage mechanisms in flexible, thin-film transistor (TFT) based Spiking Neural Networks (SNNs), and can design-time frameworks predict and mitigate these vulnerabilities?

Contribution (Chapter 5):

This chapter addresses the security of flexible electronics, a domain previously untouched by hardware security research.

FlexSpy Framework: We introduce *FlexSpy*, a unified simulation framework for assessing the SCA vulnerability of flexible neuromorphic circuits. This framework integrates a-IGZO TFT device models with circuit-level simulation to predict power side-channel signatures.

Quasi-DC Leakage: We identify a novel leakage mechanism specific to flexible SNNs: "quasi-DC" leakage. Due to the slower timescales of TFTs and the event-driven nature of SNNs, the power consumption exhibits distinct, data-dependent DC offsets correlated with the network's firing rate.

Comparative Analysis: We demonstrate that flexible SNNs (f-SNN) exhibit higher leakage potential compared to flexible (Recurrent Neural Networks (RNNs)) (f-RNN) due to the sparsity and spike-rate encoding of SNNs.

Mitigation: We propose lightweight circuit-level countermeasures, such as "event balancing," to mask the quasi-DC signature with minimal area and power overhead suitable for flexible substrates.

1.3.4. RQ4: Vulnerability of FPGA-based (HDC)

Research Question: Does the algorithmic robustness of Hyperdimensional Computing (HDC) extend to physical security, or does it mask critical vulnerabilities to deep-learning-based SCA and targeted (FI)?

Contribution (Chapter 6):

This chapter presents a rigorous real-hardware security evaluation of FPGA-based HDC accelerators.

Deep Learning SCA: We demonstrate that standard Correlation Power Analysis (CPA) is often insufficient for HDC due to the high dimensionality, but Deep Learning-based SCA (using adaptive models with ResNet-34) can successfully learn the complex leakage patterns of the XOR and Popcount units, achieving high accuracy in recovering Class Hypervectors.

Remote Sensing: We explore "Remote and Internal Sensing" attacks (e.g., "Collide & Conquer") where an adversary utilizes on-chip Time-to-Digital Converters (Time-to-Digital Converters (TDCs)) to measure voltage fluctuations on the shared PDN, enabling side-channel attacks without physical access to the board.

HyFault Injection: We introduce "HyFault," a targeted fault injection methodology that uses optimization to find the precise timing and location to inject faults that force misclassification. We show that HDC's algorithmic robustness does not protect against these worst-case, targeted perturbations.

Defenses: We develop and evaluate countermeasures tailored for HDC, including "Lightweight Dynamic Masking" and "Hypervector Randomization," which restore security with significantly lower overhead than traditional cryptographic masking.16

1.4. Methodology: Cross-Level Co-Optimization

The complexity of emerging AI accelerators demands a methodology that transcends traditional abstraction layers. A security solution optimized solely at the device level (e.g., improving the TMR ratio of MRAM) might incur unacceptable system-level energy penalties. Conversely, an algorithmic defense (e.g., boolean masking) might be ineffective if the underlying analog physics leaks information via a different channel (e.g., thermal dependence).

Therefore, this thesis adopts a Cross-Level Co-Optimization approach. This methodology explicitly models and exploits the correlations between:

Technology Level: Characterizing the fundamental physics of memory devices (ReRAM filaments, MRAM spin-torque), process variations, and thermal effects.

Circuit Level: Analyzing analog sensing margins, current summation mechanisms, PDN parasitics, and the impact of interconnect resistance.

Architecture Level: Optimizing array organization (e.g., tile size in CiM), memory hierarchy (e.g., MRAM vs. SRAM caches), and dataflow scheduling to minimize leakage exposure.

Application Level: Leveraging the properties of the AI model itself, such as the error tolerance of HDC or the sparsity of SNNs, to design algorithm-aware defenses like hypervector randomization or spike dithering.

By vertically integrating these layers, this research aims to deliver security solutions that are physically grounded, architecturally efficient, and algorithmically robust.

1.5. Summary

The shift toward post-Von Neumann computing is imperative to sustain the progress of Artificial Intelligence in the face of physical scaling limits. However, this transition cannot be made blindly. The adoption of Analog CiM, NVM, NC and HDC architectures fundamentally alters the hardware security landscape, introducing new vulnerabilities that are intrinsic to the physics of efficient computing. This dissertation provides the first comprehensive, cross-layer framework to understand, model, and mitigate these threats. By treating security as a foundational design metric co-optimized with energy and performance, this work lays the groundwork for the secure deployment of the next generation of AI hardware.

2. Background

This chapter establishes the physical and architectural ground truth that the rest of the thesis builds upon. We first motivate why, in modern AI workloads, energy and latency are dominated by *data movement* rather than arithmetic (the "memory wall"), and use this to frame two accelerator families that collapse that bottleneck: (i) memory-centric, analog (CiM) fabrics (e.g., ReRAM and STT-MRAM crossbars) and (ii) logic-centric edge accelerators (HDC and SNNs). For each case, we summarize device physics, array/microarchitectural organization, and the analog/digital non-idealities, line RC, PDN filtering, device variability, and non-volatility that shape side-channel leakage and fault susceptibility. We then formalize adversary models and attack taxonomy (observation vs. perturbation; transient vs. persistent) together with unified leakage and fault models for both digital and analog regimes, and standardize the evaluation metrics used throughout (TVLA, correlation/R^2, Signal-to-Noise Ratio (SNR), trace-to-disclosure, success rate) and the measurement stack assumptions. These definitions and primitives recur in later chapters as a common language for analyzing vulnerabilities and validating countermeasures across device, circuit, and architectural levels.

2.1. Introduction

The computational demands of modern AI, particularly in deep learning, have grown at a rate that far outpaces traditional hardware scaling. This has created an architectural crisis known as the "von Neumann bottleneck" or "memory wall" [2]. In conventional computing systems, the physical separation of logic (in the processor) and memory (in DRAM) necessitates massive, constant data movement. This data transfer is now the dominant factor in both system-level energy consumption and performance latency, especially for the data-intensive workloads of modern neural networks.

In practice, the cost to move a word from off-chip memory can exceed the cost of an entire multiply–accumulate on chip by one to two orders of magnitude, so optimizing arithmetic alone cannot recover system efficiency. Dennard scaling slow down amplifies this imbalance, and the result is that memory access patterns, interconnect distances, and power delivery limits increasingly set throughput and energy, not the raw count of arithmetic units [2], [37], [55]. This observation motivates architectures that reduce operand travel and that restructure where accumulation and nonlinearity occur in the compute path.

In response, a paradigm shift toward non-Von Neumann, domain-specific architectures is underway. This landscape is increasingly dominated by two accelerator families that seek to co-locate or merge memory and computation:

(i) **Analog, Memory-Centric Accelerators:** These platforms, often based on CiM, perform computation directly within the memory array. They exploit the physical properties of memory devices, such as Ohmic conduction ($I = V/R$) in dense (NVM crossbars (e.g., ReRAM or STT-MRAM), to execute analog MVM in-situ, thereby minimizing data movement [48], [78].

(ii) **Digital and Logic-Centric Accelerators:** These platforms map novel or bio-inspired computational models directly onto efficient hardware fabrics. Examples include HDC, which maps a symbolic algebra to digital logic [20], [89], and SNNs [5], [28], [74], which map event-driven neural computation to mixed-signal analog-digital circuits.

For memory-centric CiM, a passive crossbar implements a dot product in a single evaluate phase: row voltages encode the input vector, cell conductances encode weights, and column currents sum by Kirchhoff's current law. Column currents are then digitized by per column ADCs for downstream logic.

The benefit is that weight values do not shuttle across buses every inference step, which reduces data motion and improves effective bandwidth; the cost is sensitivity to device variability, finite dynamic range, interconnect resistance, and quantization error [48], [78].

For logic-centric designs, the mapping is algebraic. HDC represents symbols as high-dimensional random vectors, uses binding and bundling to construct composite representations, and compares classes by population count. These operations map efficiently to wide xor networks, deep carry chains, and adder trees, which is attractive for field programmable logic. SNNs implements leaky integrate and fire dynamics with current mode synapses, capacitive accumulation, and threshold comparison; spikes occur only when the membrane crosses a set threshold, so switching activity and current draw strongly follow the data stream [5], [20].

While these platforms offer compelling, and often order-of-magnitude, benefits in throughput and energy efficiency, their reliance on novel device physics, analog computation, and dense mixed-signal integration exposes a new and poorly understood class of physical attack surfaces. The very non idealities and physical phenomena (e.g., stochastic noise, process variations, interconnect parasitics) that these architectures must contend with for reliability also create vectors for insecurity.

Concretely, side channels arise because supply current and electromagnetic emissions correlate with internal states and switching activity, and fault attacks become practical when small perturbations in supply or timing alter analog thresholds or commit states. Non-volatile memories introduce persistence, so a single successful disturbance can survive resets and be reused by an adversary. These realities motivate the unified leakage and fault framework used later in this thesis, and they mirror accepted evaluation practice in hardware security, such as test vector leakage assessment for detection and persistent fault analysis for exploitation [45].

This chapter lays the architectural and physical foundations used throughout the thesis. It builds the formal models necessary to frame the subsequent security analyses of these emerging accelerators.

We proceed as follows. Section 2.2 details the device physics and circuit architectures of the accelerators central to this thesis, establishing a baseline with SRAM and then formally introducing the non-ideal physical models of ReRAM, STT-MRAM, HDC, and SNNs. The next section constructs a unified framework for physical security analysis, consolidating adversary models, attack taxonomies, formal mathematical models for side-channel leakage and fault injection, and the statistical metrics used for evaluation. We present a state-of-the-art review of physical attacks on conventional and emerging AI accelerators, synthesizing recent literature to identify the specific research gaps that this thesis addresses. Finally, Section 2.5 summarizes the chapter and provides a bridge to the novel contributions presented in subsequent chapters.

2.2. Accelerator Paradigms and Physical Foundations

This section summarizes the device physics and circuit architectures of the accelerator families analyzed in this thesis and introduces the leakage and fault primitives used later. In particular, we establish a common physical baseline—from sub-10 nm SRAM and its WL/BL parasitics to ReRAM/STT-MRAM crossbars and HDC/SNNs implementation, that explains how computation interacts with the underlying substrate. By highlighting the mechanisms through which data-dependent currents, device variability, and PDN coupling emerge, this section provides the foundations required to understand the side-channel and fault-injection behaviors.

2.2.1. Reference Platform: The Sub-10 nm SRAM Fabric

The 6-transistor (6T) Static Random Access Memory (SRAM) bitcell in Fig. 2.1 is the foundational storage element in modern digital logic. It comprises two cross-coupled inverters (a bistable latch) and two access transistors that connect the internal storage nodes to the differential bitline (BL) pair under wordline (WL) control. The cell resides in a macro with peripheral address decoding, sensing, precharge, and timing control. In advanced designs, the bitcell must also balance read stability against write ability

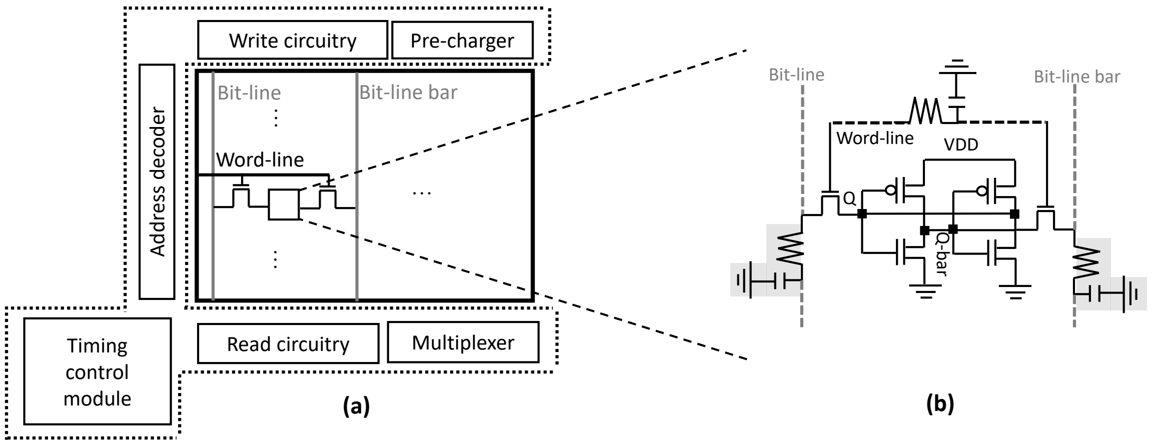


Figure 2.1.: (a) SRAM macro with periphery; (b) 6T bitcell. Distributed WL/BL RC parasitics degrade read/write margins and shape dynamic current signatures.

by carefully sizing the pull-up and pull-down devices, ensuring that the internal nodes do not inadvertently flip during read-back. Furthermore, sense-amplifier and precharge circuits introduce additional analog dynamics, such as offset, meta-stability sensitivity, and timing skew, that influence the instantaneous supply current during memory accesses. These second-order effects become relevant for later sections that compare SRAM behavior to emerging CiM fabrics.

While SRAM is logically digital, advanced sub-10 nm nodes reintroduce analog challenges. As interconnect cross-sections shrink, WL/BL resistance dominates, creating IR drops that reduce static noise margin (Static Noise Margin (SNM)) and stress read, write, and hold operations.[1] Accurate models therefore treat WL/BL not as ideal wires but as distributed RC networks discretized at cell pitch. In addition, capacitive coupling between adjacent bitlines and wordlines introduces data-dependent disturbances that influence both access latency and the transient current profile. Variability from random dopant fluctuations and line-edge roughness further broadens the distribution of cell margins, making the access behavior increasingly sensitive to Process, Voltage, and Temperature (PVT) conditions. This analog-like behavior matters twice in this thesis: (i) as a *baseline memory fabric* against which CiM is compared, and (ii) because SRAM-based CiM primitives (e.g., bitline computing) inherit the same non-idealities (local mismatch, IR drop).

2.2.2. Memory-Centric Accelerators: Analog Compute-in-Memory (CiM)

Analog CiM performs matrix–vector multiplication (MVM) directly in a memory crossbar by exploiting Ohm's and Kirchhoff's laws. For input row voltages $\boldsymbol{v} \in \mathbb{R}^N$ and synaptic weights stored as crosspoint conductances $\boldsymbol{G} \in \mathbb{R}^{N\times M}$, the column-$j$ current is

$$i_j(t) = \sum_i G_{ij}\, v_i(t) \qquad \Rightarrow \qquad \boldsymbol{i}(t) \approx \boldsymbol{G}^\top \boldsymbol{v}(t), \tag{2.1}$$

which is then digitized by per-column ADCs. The promise of CiM relies on dense NVM devices as programmable conductances G_{ij}. In practical crossbars, this current accumulation is also shaped by access device resistance, finite line impedance, and bitline capacitance, which together introduce RC-induced dispersion in the temporal response. The effective conductance matrix therefore, depends on both the programmed cell states and parasitic elements distributed across the array, causing input-dependent attenuation or distortion in $i_j(t)$. Furthermore, variations in device switching behavior and nonlinearity in the cell $I-V$ characteristics create additional offsets in the accumulated current, making these analog dynamics a key origin of leakage and fault behavior studied later in this thesis.

[1] In practice, write margins tend to be most sensitive because the WL driver must overcome line resistance to enable the access devices.

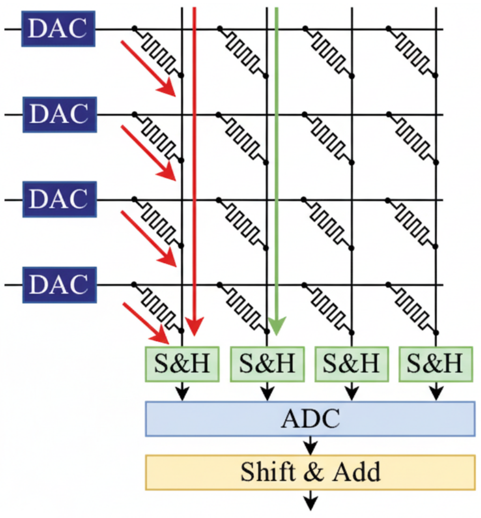


Figure 2.2.: Architecture of Compute-in Memory (CiM).

2.2.2.1. Device Physics I: Resistive RAM (ReRAM)

ReRAM stores information in the resistance of a metal–insulator–metal stack. Switching is filamentary; an applied field forms or ruptures a nanoscale conductive filament in the insulator. In oxide-based devices (VCM/OxRAM), oxygen ion migration creates or erases an oxygen-vacancy filament, yielding low- and high-resistance states (G_{LRS}, G_{HRS}). A key security implication is that filament formation is *stochastic*, producing (i) device-to-device and cycle-to-cycle variability in $G_{\mathrm{LRS}}/G_{\mathrm{HRS}}$, and (ii) Random Telegraph Noise (RTN) during reads, as carriers trap or de-trap in CF defects.

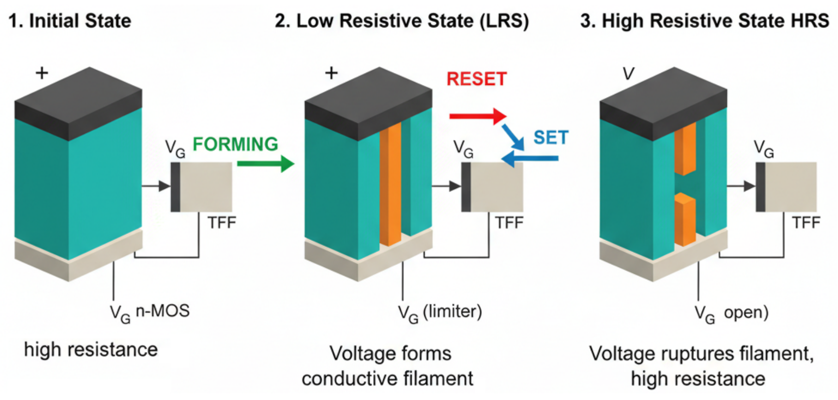

1. Initial State
2. Low Resistive State (LRS)
3. High Resistive State HRS
+
VG FORMING
TFF
VG n-MOS
high resistance
+
RESET
SET
VG
TFF
VG (limiter)
Voltage forms conductive filament
V
VG
TFF
VG open)
Voltage ruptures filament, high resistance

Figure 2.3.: Structure and operation of Resistive-Random Access Memory (ReRAM).

In addition, the nonlinear $I-V$ characteristics of the CF at low voltages introduce bias-dependent conductance shifts that modulate the readout current in a data-dependent manner. Local self-heating in the filament can further aggravate temporal fluctuations, especially during consecutive accesses or high-frequency operation. The surrounding access transistor and line parasitics also interact with the intrinsic filament dynamics, shaping the instantaneous current seen at the bitline. These combined effects propagate upward through the array and PDN, creating distinct analog signatures that become exploitable leakage sources in later chapters.

2.2.2.2. Device Physics II: Spin-Transfer-Torque MRAM (STT-MRAM)

STT-MRAM uses a 1T1MTJ cell whose resistance depends on the relative magnetization of a pinned and a free ferromagnetic layer, separated by MgO: R_P (parallel) vs. R_{AP} (anti-parallel), with TMR = $(R_{AP} - R_P)/R_P$. Writes employ spin-polarized current; the resulting spin-transfer torque flips the free layer if the current–time integral exceeds a critical threshold. Three features are security-relevant: (i) *asymmetric* switching (P→AP vs. AP→P) with different critical conditions, (ii) *stochastic* switching (thermal assistance), and (iii) *read disturb* risk if the read current biases the MTJ too aggressively.

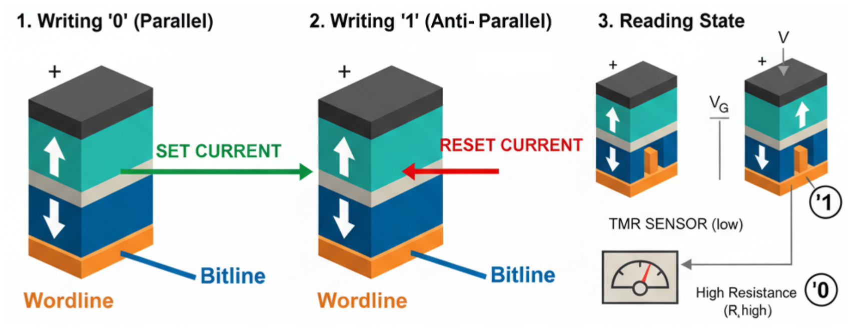


Figure 2.4.: Structure and operation of Spin Torque Transfer-Magnetic Random Access Memory (STT-MRAM).

In addition, the MTJ resistance shows bias-dependent nonlinearity and voltage-controlled interfacial anisotropy, which slightly shifts R_P and R_{AP} during high-frequency accesses. The thermal stability factor of the free layer impacts both switching probability and retention time, creating small but measurable variations in read current that manifest as low-frequency noise at the array level. Line resistance and access transistor variations also modulate the effective write current delivered to the MTJ, creating spatially varying vulnerability to commit window faults. These combined effects become important when modeling persistent faults and supply-coupled leakage in later chapters.

2.2.2.3. CiM Architecture and Analog Non-Idealities (ANIs)

STT-MRAM-based CiM. Under soft-read bias ($\ll I_c$), an MTJ behaves as a state-dependent resistor:

$$I_{ij}(t) = \frac{V_{ij}(t)}{R_t + R_{ij}}, \quad R_{ij} = R_P + b_{ij}(R_{AP} - R_P), \quad b_{ij} \in \{0, 1\}, \tag{2.2}$$

with R_t capturing access/line parasitics. Non-idealities include bias-dependent read disturb, thermal magnetization noise (timing jitter), and mild voltage non-linearity.

ReRAM-based CiM. Reads follow $I_{ij}(t) = V_{ij}(t)/(R_t + R(b_{ij}, t))$, where $R(b_{ij}, t)$ is stochastic due to D2D variability and RTN. Additional ANIs include IR drops from line resistance, device I–V non-linearity (e.g., $I = G_0V + \eta V^3$), and sneak paths.

Power, PDN, and the leakage model. With ANIs, the conductance matrix becomes time-varying: $G_{\text{actual}}(t) = G_{\text{ideal}} + \Delta G(t)$. Let $I_{\text{array}}(t) = \sum_j i_j(t)$ be the array current. The supply-observed current is the PDN-filtered version plus measurement noise:

$$I_{\text{VDD}}(t) = [I_{\text{array}}(t) * h_{\text{PDN}}(t)] + n(t). \tag{2.3}$$

Instantaneous power is therefore

$$P(t) \approx \boldsymbol{v}(t)^\top G_{\text{actual}}(t)\,\boldsymbol{v}(t) = \underbrace{\boldsymbol{v}^\top G_{\text{ideal}}\boldsymbol{v}}_{\text{ideal MVM}} + \underbrace{\boldsymbol{v}^\top \Delta G(t)\boldsymbol{v}}_{\text{leakage term}}. \tag{2.4}$$

The second term quantifies the physics-driven side-channel: leakage scales with the correlation between inputs $\boldsymbol{v}$ and non-idealities $\Delta G(t)$. In practice, $h_{\text{PDN}}(t)$ introduces attenuation, ringing, and low-pass behavior determined by on-die and package-level decoupling, which distort the observable current signature in a repeatable but circuit-dependent manner. Spatial coupling among wordlines and bitlines also causes correlated fluctuations in $I_{\text{array}}(t)$, especially when multiple columns evaluate concurrently. Furthermore, device-level fluctuations such as RTN or thermal drift alter $\Delta G(t)$ across repeated queries, creating small but statistically exploitable variations in $P(t)$ that reappear in later security analysis.

Compute primitives. CiMs supports various logic computing such as: (i) analog MVM/MAC with post-activation (e.g., translinear tanh), (ii) in-sense window-comparator logic (e.g., scouting XOR), and (iii) CAM-like match-line discharge (similarity search, discharge $\propto$ Hamming distance). These will be targets for the security analyses that follow. Each primitive exercises different parts of the array and periphery, resulting in distinct temporal current profiles that pass through the same PDN. For example, MAC operations emphasize continuous current accumulation, whereas comparator-based logic produces short, high-slope transients tied to decision thresholds. CAM style operations heavily stress match-lines and discharge paths, creating characteristic exponential decay signatures on the supply rail. These variations enable primitive specific leakage models that are later used to quantify vulnerability and estimate traces to disclosure.

2.2.3. Hyperdimensional Computing (HDC) on FPGA/ASIC

HDC is an alternative computational paradigm inspired by the distributed, holographic encoding observed in biological neural systems. Instead of operating on low-dimensional numeric values, HDC manipulates high-dimensional binary or bipolar hypervectors whose redundancy provides natural robustness, rapid learning, and efficient hardware realizations. The algebra of binding, bundling, and similarity enables symbolic reasoning and classification using simple bitwise and arithmetic primitives. These properties make HDC particularly attractive for FPGAs and Application-Specific Integrated Circuits (ASICs) accelerators where massively parallel switching and deterministic dataflow can be exploited for high throughput and energy efficiency.

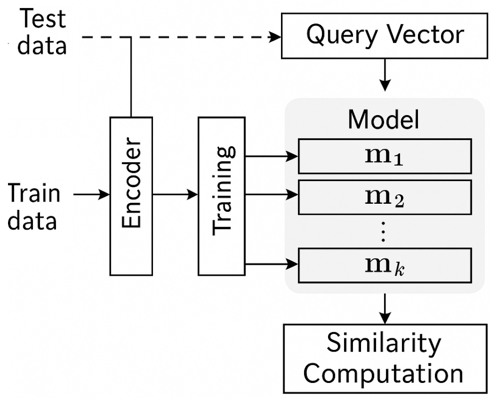


Figure 2.5.: Overview of different operations in Hyperdimensional Computing (HDC).

HDC represents symbols as high-dimensional hypervectors $\boldsymbol{h} \in \{\pm 1\}^D$ or $\{0, 1\}^D$ (e.g., $D = 10,000$) and relies on three operations: binding (XOR for binary), bundling (element-wise majority/threshold), and similarity (dot product/Hamming distance):

$$S_c = \boldsymbol{h}_q^{\top} \boldsymbol{w}_c, \qquad \hat{c} = \arg\max_c S_c. \tag{2.5}$$

On FPGAs/ASICs, binding maps to LUT trees; bundling and similarity map to adder trees. On FPGAs, synthesis targets dedicated carry chains (for example, `CARRY4`) for speed, concentrating data-dependent switching there. This microarchitectural detail dominates dynamic power and becomes the primary source of high-SNR leakage exploited later.

In addition, wide XOR and popcount structures switch thousands of bits per cycle, creating large, temporally aligned current bursts that are highly correlated with the Hamming distance in equation (2.5). The routing fabric and carry chain segmentation also introduce small but measurable timing skew that accentuates these correlation peaks on the supply rail. BRAM readout of stored hypervectors further adds deterministic access patterns that serve as reliable alignment markers in power and timing traces. These architectural traits collectively define the leakage and fault surfaces that Chapters 5 and 6 exploit and mitigate.

2.2.4. Spiking Neuromorphic Computing (NC) on Emerging Technology

Spiking Neural Networks (SNNs) provide a bio-inspired computational model for neuromorphic computing (NC), in which information is represented and processed through discrete spike events instead of continuous activations. This event-driven nature reduces redundant switching activity and enables energy-efficient inference, making SNNs suitable for resource-constrained edge platforms and emerging substrates such as flexible electronics. Hardware implementations often combine analog membrane dynamics with digital spike generation, resulting in a mixed-signal pipeline with characteristic timing and current signatures. These properties introduce unique leakage and fault behaviors that differ significantly from conventional ANN accelerators.

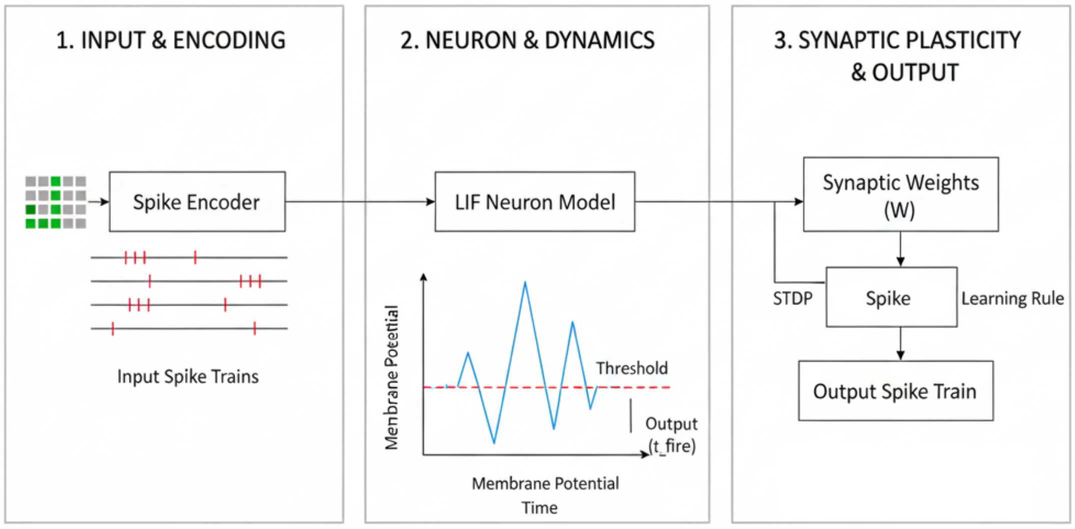


Figure 2.6.: Overview of different operations in spike-based Neuromorphic Computing (NC).

SNNs communicate via discrete spikes. Analog implementations of Leaky Integrate-and-Fire (LIF) neurons comprise: (i) synaptic integration (Digital-to-Analog Converters (DACs) or transconductors into a membrane capacitor C_{mem}), (ii) a leak path, and (iii) a comparator that fires when $V_{\text{mem}} > V_{\text{th}}$ and then resets. Two security-relevant aspects follow. First, the analog front-end (DAC or Transimpedance Amplifier (TIA) and associated biasing) has characteristic transient responses and current draw during spike epochs, creating repeatable power-leakage windows. Second, V_{th} is often derived from V_{DD} (for example, a resistor divider), so supply glitches directly modulate the effective threshold, providing a physically simple fault-injection path that advances or delays the firing moment.

In addition, the membrane integration process introduces low-frequency components in the supply current that scale with aggregate synaptic activity, making spike-rate-dependent leakage particularly strong. The reset event typically produces sharp current edges that serve as reliable temporal markers in power traces. Device-level non-idealities in TFT or CMOS neuron circuits, such as leakage currents or temperature-dependent bias drift, further modulate the firing behavior in a data-correlated manner. These effects collectively define the analog signatures that the flexible neuromorphic security analysis exploits in later chapters.

2.3. Adversary Models and Attack Taxonomy

The feasibility and power of a physical attack are primarily determined by the adversary's access and capabilities. We consider three adversary models:

1. **Laboratory Adversary (white box):** Common in side channel and fault injection studies [4], [6], [16], [18], this adversary has full physical possession of the device (e.g., a test chip or edge board).

They can depackage and probe the die and deploy high-end instrumentation (oscilloscopes, probes, pulse generators). Invasive and semi-invasive techniques are in scope.

2. **Multi tenant Adversary (gray box):** Relevant to cloud FPGA platforms [62], [64]. The attacker lacks physical access but can co-locate a malicious *spy* design alongside a *victim* on the same fabric. Attacks proceed by exploiting shared resources, for example, the on-chip PDN, clock trees, or on-chip interconnects (Advanced eXtensible Interface (AXI)), to sense or perturb the victim remotely.

3. **Supply chain Adversary (white box at design/fab):** Intersects design and manufacturing. The adversary can insert a *Hardware Trojan* or leverage process non-idealities (e.g., in memristor fabrication) to create backdoors or reliability-driven failures [29].

Unless stated otherwise, device-level characterization in this thesis assumes the Laboratory Adversary.

Attack Taxonomy. We classify attacks along two axes: (i) *observational* (measuring) vs. *perturbational* (injecting), and (ii) *transient* vs. *persistent*. Table 2.1 summarizes the categories used throughout.

Table 2.1.: Physical attack taxonomy used in this thesis.

Category	Mechanism (examples)	Effect type	Typical toolchain
Side-Channel (SCA)	Supply current / EM / timing observations	Observational; transient	Oscilloscope, Integrated Logic Analyzer (ILA), Electromagnetic (EM) probe; trigger
Fault Injection (FI)	Voltage/clock glitch; EM/laser pulse; PDN droop	Perturbational; transient	ChipWhisperer, pulse generator
Persistent Faults	MRAM commit-window corruption; ReRAM drift/retention	Perturbational; persistent	Rail crowbar switch, scripted cycling

2.3.1. Formal Models of Physical Attack Vectors

We formalize the attack surfaces most relevant to this work: side channel leakage (observation) and fault injection (perturbation). These models bridge device-level behavior with circuit and architectural execution, allowing us to express how physical events map to measurable or exploitable system responses. For leakage, we characterize how data-dependent currents, switching activity, and PDN coupling produce statistically distinguishable features under realistic noise. For fault injection, we describe how timing, voltage, or device state disturbances propagate through mixed-signal and digital pathways to alter computation. These formal definitions provide the analytical foundation required for the vulnerability assessments carried out in later chapters.

2.3.1.1. Side-Channel Leakage Models (Observation)

Side channel attacks exploit data-dependent physical emissions. The two platforms under study, synchronous digital logic (FPGA/ASIC) and analog CiM, exhibit fundamentally different leakage mechanisms. These mechanisms arise from distinct physical origins, timing structures, and noise behaviors, which must be captured by separate leakage models. Expressing both within a unified notation allows a coherent comparison across architectures and enables cross-technology vulnerability analysis that is central to this thesis.

Digital (CMOS) Leakage Model. In synchronous CMOS, power is dominated by *dynamic* switching of load capacitances $C_{L,k}$ at logic nodes k [6], [10], [12], [18], [24]:

$$P_{\text{CMOS}}(t) \approx \sum_k \alpha_k(t)\, C_{L,k}\, V_{DD}^2 f \;+\; P_{\text{static}} \;+\; \varepsilon(t), \tag{2.6}$$

where f is the clock frequency, P_{static} accounts for leakage, $\varepsilon(t)$ models noise, and $\alpha_k(t) \in \{0,1\}$ is the activity factor (whether node k switched at time t). For a D-bit register update, aggregate switching correlates with the *Hamming distance* between old and new values [12], [18], forming the basis of Correlation Power Analysis (CPA) used against digital logic, including the HDC accelerators in Chapter 6. In practice, routing capacitances, clock tree loading, and glitching effects in combinational paths further modulate $\alpha_k(t)$, creating fine-grained temporal features in $P_{\text{CMOS}}(t)$. These secondary effects act as alignment cues in real measurements and often strengthen information leakage.

Analog (CiM) Leakage Model. In analog CiM, leakage stems from *the analog current summation itself*, which dominates CMOS switching. Using the power model in Eq. 2.4,

$$P(t) \approx \boldsymbol{v}^\top \boldsymbol{G}_{\text{ideal}}\, \boldsymbol{v} \;+\; \boldsymbol{v}^\top \Delta \boldsymbol{G}(t)\, \boldsymbol{v},$$

the observed supply current $I_{\text{VDD}}(t)$ follows the PDN convolution of Eq. 2.3 [48], [63], [70], [78]. Leakage therefore depends on the *analog values* of inputs $\boldsymbol{v}$ and conductances $\boldsymbol{G}$, including non-idealities $\Delta\boldsymbol{G}(t)$. Unlike CMOS switching, this leakage is continuous in amplitude and persists over the full evaluate window, making averaging highly effective for weight extraction. Furthermore, parasitic resistances and device noise introduce structured distortions that remain correlated across repeated queries, enabling attack models that exploit temporal and spectral features of the current waveform.

Quantifying Leakage. Leakage strength is assessed via the Pearson correlation ρ between N observed features (e.g., windowed power) L_{obs} and N model predictions L_{model}:

$$\rho \;=\; \frac{\text{Cov}(L_{\text{obs}}, L_{\text{model}})}{\sqrt{\text{Var}(L_{\text{obs}})\,\text{Var}(L_{\text{model}})}}. \tag{2.7}$$

In digital settings L_{model} may be Hamming weight or distance; in CiM, it corresponds to analog MAC-derived predictions. Correlation is particularly useful because it is agnostic to absolute scale and captures linear dependence even under PDN filtering and moderate noise. Additional metrics, such as R^2 and SNR, complement ρ and appear later in later Chapters to rigorously compare leakage across architectures and PVT corners.

2.3.1.2. Fault Injection Models (Perturbation)

Fault injection perturbs operating conditions to induce computational errors. We model three attack classes. These models capture how device-level variations, circuit delays, and PDN dynamics translate into erroneous architectural behavior that can be exploited by an adversary. Expressing these mechanisms formally allows quantitative reasoning about error rates, sensitivity windows, and the operating regimes in which attacks become practical.

Digital Transient Faults (Timing Violation). Synchronous logic assumes timing closure. A setup time violation occurs when the data at a flip-flop input arrives after its setup window relative to the capturing edge. Static Timing Analysis (STA) requires [30], [33]:

$$T_{clk} \;\geq\; D_{clk2q} + D_{pMax}(V_{DD}, T) + T_{setup} - T_{skew}, \tag{2.8}$$

where D_{pMax} (max path delay) is strongly dependent on supply V_{DD} and temperature T. An attacker can violate Eq. 2.8 by:

- **Clock glitching:** transiently shortening T_{clk},
- **Voltage glitching:** inducing a droop ΔV_{DD} that increases D_{pMax}.

Resulting faults include *metastability*, non-deterministic behavior, or *early latching*, deterministic re-latching of the prior value [16]. Additional timing hazards, such as logic hazards, unequal path skews, and temporary pulse filtering, broaden the vulnerability window, especially in deeply pipelined datapaths. The PDN also couples disturbances across multiple logic blocks, amplifying the practical reach of a single glitch event.

Transient Faults (Threshold Shifting). In analog SNNs, the analog comparator threshold V_{th} is often derived from V_{DD} (Section 2.2.4). A voltage glitch ΔV_{DD} therefore perturbs the decision threshold:

$$V_{\text{th,faulty}}(t) = V_{\text{th,ideal}} + f\left((\Delta V_{DD} * h_{\text{PDN}})(t)\right), \tag{2.9}$$

with $f(\cdot)$ approximating the threshold droop transfer [28]. The LIF neuron then fires early or late, producing transient computational errors. Because the membrane voltage integrates current over time, even a brief droop introduces cumulative deviation, shifting the firing moment by several milliseconds. Comparator offset, bias current variation, and device leakage further magnify the effect, creating a wide attack surface for low-cost voltage fault injection.

Persistent Faults (NVM State Corruption). Non-volatile accelerators admit faults that *survive power cycles*.

- **STT-MRAM:** Exploits the *asymmetric, stochastic write process* (Section 2.2.2.2). A pulse ($V_{\text{pulse}}, t_{\text{pulse}}$) can be chosen to reliably flip AP→P but not P→AP [35], yielding data dependent reset to one or reset to zero faults. The write path's sensitivity to current density and thermal fluctuations creates a narrow but repeatable commit window that an attacker can target with nanosecond precision. Once implanted, such faults remain stable until explicitly rewritten, enabling repeated exploitation.
- **ReRAM:** Repeated sub-threshold stress can induce cumulative resistance drift and retention changes, corrupting analog weights [38], [63]. Local Joule heating, ion migration, and stochastic filament reshaping cause gradual but persistent shifts in G_{LRS} or G_{HRS}, which distort analog MAC outputs long after the initial perturbation. These effects represent a long-lived fault channel distinct from the transient behavior seen in CMOS or SRAM.

2.3.1.3. Evaluation Methodology and Metrics

We standardize metrics from detection to exploitation and detail the measurement stack used across experiments. These metrics provide a common basis for comparing leakage and fault sensitivity across the diverse substrates analyzed in this thesis, including CMOS, analog CiM, flexible neuromorphics, and FPGA HDC. They capture not only whether leakage is present but also how strongly it correlates with secret data, how many traces or injections are needed for successful exploitation, and how reproducible an induced fault is under realistic noise. Establishing these quantitative measures enables rigorous, cross-chapter evaluation of attack feasibility and countermeasure effectiveness.

- **Welch's t-test (TVLA):** Used for *leakage detection*. TVLA tests the null hypothesis that two trace populations, for example fixed vs. random inputs, share the same mean [24], [26], [45]. Welch's two-sample statistic is robust to unequal variances. The industry threshold $|t| > 4.5$ indicates rejection of H_0 with very high confidence [45].
- **Correlation ρ and explained variance R^2:** Quantify the fit of a *specific* leakage model, digital or analog. With $R^2 = \rho^2$, higher values imply a larger share of explained trace variance [18].

- **SNR:** Separability of data dependent signal from noise, SNR = $\frac{\mathrm{Var}[\mathbb{E}[P|x]]}{\mathbb{E}[\mathrm{Var}[P|x]]}$, with larger SNR indicating an easier attack [8], [14], [24].
- **Success Rate (SR):** Empirical probability of correct recovery, key, weight, class, as a function of traces or injected faults.
- **Trace to Disclosure (N_{TD}):** Practicality estimate for correlation based attacks. For target confidence z and dominant correlation $\rho_\star$, $N_{\mathrm{TD}} \approx z^2/\rho_\star^2$, assuming trace independence.
- **Persistence score (π):** For Chapter 3. Fraction of induced faults surviving n power cycles, $\pi \approx 1$ indicates a truly persistent MRAM or ReRAM fault.

2.4. Identified Research Gaps and Thesis Positioning

This comprehensive review of the state of the art reveals clear (as shown in Table 2.2) and critical research gaps. The vast majority of existing security analysis focuses on digital implementations of neural networks, on CMOS, FPGAs, and GPUs, and employs digital leakage and fault models, for example Hamming weight and logical bit flips [18], [24]. These models are well-suited for synchronous switching activity in standard logic, yet they do not capture the physics that govern computation in emerging accelerators, where current summation, device variability, and analog front ends dominate the signal.

Table 2.2.: Attacks and defenses on emerging AI accelerators arranged by *Impact* (rows) and *Access* (columns). Red bullets denote *attacks*; blue bullets denote *defenses*. Technology tags: CiM = compute-in-memory, SNN = spiking neural networks, HDC = hyperdimensional computing.

Impact / Access	Training-time	Remote	Invasive Physical
Full System Compromise	● *Data poisoning* (CiM, SNN) [31] ● *Backdoors / Neural Trojans* (CiM, SNN) [51, 66]	● *Redundancy verification* (e.g., TMR) [1]	● *Shielding & redundancy* [3, 15]
Local User Privacy Violation	● *Membership inference* (CiM, SNN, HDC) [54]	● *Model extraction* (SNN) [49]	● *Microprobing protection (active shields, sensors)* (CiM, sensors) [15]
Model Privacy Violation	● *Side-channel leakage* (CiM) [6, 18]	● *Adversarial queries / inference* [53]	● *Power/clock injection detection* (EMFI/clock) [32, 34]
Denial of Service	● *Constant-activity neurons* (SNN) [13]	● *Packet flooding / activity saturation* [13]	● *Glitch detectors, shielding* [34]

Legend: ● Attack ● Defense

In contrast, the security of the emerging analog, memory-centric, and logic-centric, HDC, paradigms is identified by numerous sources as a major open problem [63], [78], [89]. The unique vulnerabilities introduced by NVM device physics, analog computation, and novel microarchitectures remain largely unexplored. The central, open questions are:

1. **Gap 1, Analog SCA.** How do the fundamental analog non idealities of CiM, such as ReRAM random telegraph noise and device to device variability, and STT-MRAM thermal noise, translate into

exploitable side channel leakage? Existing digital, Hamming weight models are insufficient because they do not describe the correlation between input voltages and the time-varying conductance matrix that arises from physical noise and process variation. A new, physics based analog leakage model is required, one that can be instantiated on specific analog compute primitives such as scouting logic XOR, MAC plus tanh activation, and CAM style similarity search, and that explicitly links $\boldsymbol{v}$, $\boldsymbol{G}$, and the measured supply signature through the PDN model introduced earlier [63], [70], [78].

2. **Gap 2, Physics-aware and microarchitecture-aware attacks.** How can an attacker exploit the unique device physics of NVMs and the specific microarchitecture of digital accelerators to mount more powerful fault attacks and higher fidelity leakage analyses?

   (a) **Persistent faults.** While PFA and STT-MRAM write vulnerabilities, for example, asymmetric switching and stochastic commit windows, are known independently, no prior work combines them to demonstrate a practical, physics-based persistent fault attack that targets the MRAM commit window and yields data-dependent stuck-at behavior across power cycles [35].

   (b) **Digital edge AI.** The security of HDC on FPGAs is under studied. Practical, real hardware attacks that target the specific HDC microarchitecture, for example, the carry chain-based popcount and adder trees that dominate power, using advanced remote sensing, TDC, and learning based analysis have not been demonstrated [56], [57], [89].

2.5. Summary

This chapter established the conceptual and physical foundations required for the remainder of the thesis. We first motivated why modern AI workloads are increasingly bottlenecked by data movement and outlined how emerging non–Von Neumann architectures, particularly analog CiM, neuromorphic computing, and HDC, seek to overcome this limitation. We then introduced the device-level physics of ReRAM, STT-MRAM, TFT-based flexible electronics, and advanced CMOS fabrics, emphasizing how their non-idealities influence computation, power consumption, and vulnerability to physical attacks.

A unified threat model was developed, covering observational (SCA), perturbational (FI), transient, and persistent fault scenarios. We formalized leakage and fault models for both digital and analog compute substrates, and defined statistical metrics such as TVLA, correlation, SNR, mutual information, and persistence scores that will be used consistently throughout the thesis to evaluate security.

The subsequent chapters address the above-mentioned gaps directly: Chapter 3 develops the analog leakage framework for CiM, Chapter 4 demonstrates persistent-fault exploitation in MRAM devices and its cryptographic implications, Chapter 5 explores a framework for side-channel security verification before manufacture, and Chapter 6 applies both external and internal side-channel methodologies to FPGA-based HDC accelerators.

Part II.

Contributions

3. Vulnerabilities of Emerging NVM-based In-memory Computing: Simulation

3.1. Introduction

The rapid growth of data-intensive workloads—especially Large Language Models (LLMs) and generative AI—has exposed the limits of von Neumann systems. When processing and memory are separated, data movement dominates both latency and energy (often by 10×-100× over arithmetic), creating the well-known "memory wall." Compute-in-Memory (CiM) mitigates this gap by performing operations *in* or *near* the array, improving energy efficiency and throughput for Machine Learning (ML)/Deep Neural Network (DNN) inference [2], [59], [63], [78].

This chapter focuses on **Analog CiM** in contrast to Digital processor-centric design. Processor-centric architecture largely relocates CMOS logic close to memory arrays; Analog Compute-in-Memory (ACIM) instead exploits device physics (e.g., ReRAM, STT-MRAM) to perform MAC via Ohm's law and Kirchhoff's Current Law (KCL). In many ACIM designs, model parameters are programmed as fixed cell conductances and on-chip readout prevents direct weight dumps, creating a software-level "black box" [42], [63], [78].

Security-relevant consequence. The same black-box property typically makes weights stationary physical states bound to specific cells. An adversary who can issue repeated queries and observe power or EM signals can average measurements, suppressing uncorrelated noise roughly as $1/\sqrt{N}$ over N traces. Thus, software opacity can inadvertently help SCA: the target is stationary, and the adversary can shape inputs to probe it. Moreover, the spatial distribution of programmed conductances creates characteristic current pathways that remain identical across evaluations, producing repeatable analog signatures even under moderate measurement noise. Bias-dependent non-idealities in the devices further amplify these correlations, making leakage more distinguishable than in deeply pipelined CMOS logic. Combined, these factors convert the physical array structure itself into a stable oracle that an attacker can exploit through carefully crafted input ensembles.

From discrete to continuous leakage. Unlike digital SCA—which relies on discrete switching activity and Hamming-style models—ACIM leakage is the superposition of continuous analog currents and array parasitics, subsequently filtered by the power-delivery network (PDN) and measurement path [6], [8], [12], [19], [27]. A minimal readout model that we will use throughout the chapter is

$$I_{\mathrm{BL}}(t) = \sum_i V_i(t)\, G_i \; + \; I_{\mathrm{par}}(t) \; + \; \eta_{\mathrm{dev}}(t), \tag{3.1}$$

$$z(t) = (h * I_{\mathrm{BL}})(t) \; + \; n_{\mathrm{meas}}(t), \tag{3.2}$$

where $V_i(t)$ are input voltages, G_i are stationary cell conductances (weights), $I_{\mathrm{par}}(t)$ captures line/selector leakage and RC dynamics, $\eta_{\mathrm{dev}}(t)$ models device noise (e.g., thermal and RTN), h is the PDN/analog front-end impulse response, and $n_{\mathrm{meas}}(t)$ is measurement noise. Equations (3.1)-(3.2) highlight three practical differences: (i) leakage is continuous-valued and input-weighted, (ii) it is spatio-temporal due to IR/RC and PDN filtering, and (iii) it benefits from repeated-query averaging because G_i are time-invariant during an attack.

Table 3.1 formalizes these contrasts and the resulting attack surface.

Table 3.1.: Side-channel paradigms: Digital CMOS vs. Analog CiM.

Feature	Digital CMOS (Conventional)	Analog CiM (Emerging)
Leakage source	Gate switching (dynamic power)	Analog current aggregation in crossbars (Ohm, KCL)
Leakage nature	Discrete, data-dependent (Hamming weight/distance)	Continuous; $\sum_i V_i G_i$ with parasitic terms
Time structure	Localized in cycles/events	Dispersed; shaped by RC/IR drop and PDN filter h
Dominant noise	Process, quantization, background	Device noise (e.g., RTN), thermal, parasitic distortions
State persistence	Transient internal states	Stationary conductances (weights) during inference
Averaging leverage	Limited—state changes frequently	High—stationary weights enable $1/\sqrt{N}$ averaging
Typical attacks	CPA, Differential Power Analysis (DPA), simple templates	Profiled templates (Mahalanobis), Principal Component Analysis (PCA)/denoising, regression
Primary target	Cryptographic keys/state bits	Model architecture, weights, and sometimes inputs
Measurement focus	Core/logic power rails	Array rails, bitline current, EM from word/bitlines

Implication for this chapter. Because ACIM leakage encodes $V \times G$ interactions and is shaped by analog dynamics, effective attacks must (i) preserve or reconstruct the PDN/measurement filter h, (ii) excite the array with inputs that disambiguate conductances, and (iii) exploit stationarity via averaging and template based inference. The remainder of the chapter develops a simulation-driven attack framework around (3.1)-(3.2). In addition, the temporal alignment of evaluate and sense phases introduces structured windows in which the leakage model is most predictive, allowing focused feature extraction rather than full trace analysis. Device-level noise sources, while present, remain correlated with operating conditions and therefore contribute usable statistical signatures. These properties make analog CiM particularly suitable for profiling-based attacks, which benefit from the deterministic and repetitive nature of the underlying physical computation.

3.2. Physical Origins of Analog Leakage: From Device Physics to Circuit Dynamics

We construct a bottom-up threat model that links device physics to supply-rail/EM emissions through three layers:

1. **Device (D)**: Bitcell read currents in ReRAM and STT-MRAM. At this level, the physical state of each memory element is represented as a conductance value subject to device-to-device variability, temporal noise processes, and bias-dependent nonlinearity. ReRAM cells exhibit filamentary randomness and RTN, while STT-MRAM cells show thermal activation and asymmetric resistance distributions tied to magnetization states. These effects directly determine the instantaneous cell current under a given read bias and become the earliest source of information leakage. Furthermore, device physics governs the sensitivity to perturbation, for example read disturb in ReRAM or commit window vulnerability in MRAM, which later translates to exploitable attack vectors.

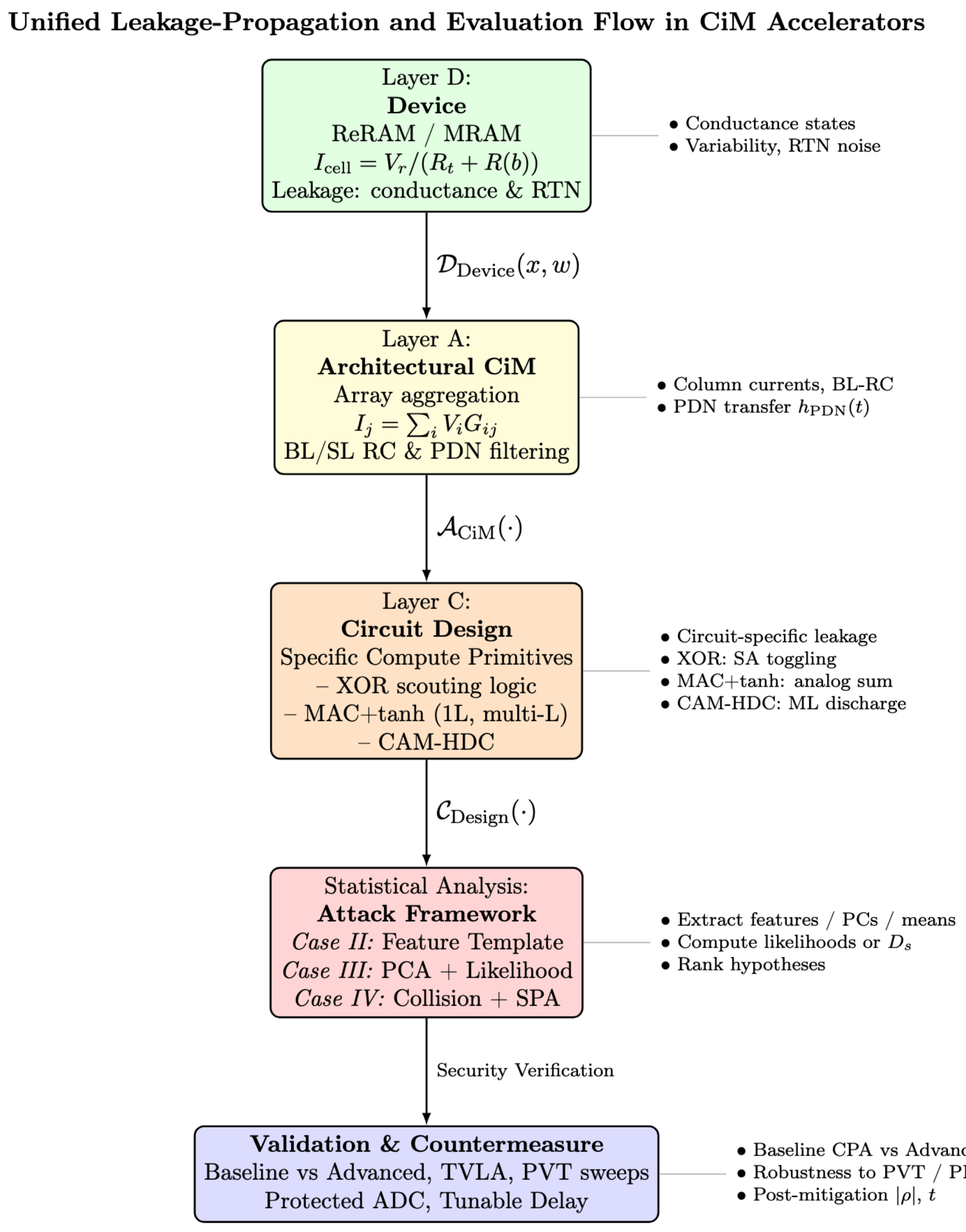

Unified Leakage-Propagation and Evaluation Flow in CiM Accelerators
Layer D:
Device
ReRAM / MRAM
$I_{\text{cell}} = V_r/(R_t + R(b))$
Leakage: conductance & RTN
• Conductance states
• Variability, RTN noise
$\mathcal{D}_{\text{Device}}(x, w)$
Layer A:
Architectural CiM
Array aggregation
$I_j = \sum_i V_i G_{ij}$
BL/SL RC & PDN filtering
• Column currents, BL-RC
• PDN transfer $h_{\text{PDN}}(t)$
$\mathcal{A}_{\text{CiM}}(\cdot)$
Layer C:
Circuit Design
Specific Compute Primitives
– XOR scouting logic
– MAC+tanh (1L, multi-L)
– CAM-HDC
• Circuit-specific leakage
• XOR: SA toggling
• MAC+tanh: analog sum
• CAM-HDC: ML discharge
$\mathcal{C}_{\text{Design}}(\cdot)$
Statistical Analysis:
Attack Framework
Case II: Feature Template
Case III: PCA + Likelihood
Case IV: Collision + SPA
• Extract features / PCs / means
• Compute likelihoods or D_s
• Rank hypotheses
Security Verification
Validation & Countermeasure
Baseline vs Advanced, TVLA, PVT sweeps
Protected ADC, Tunable Delay
• Baseline CPA vs Advanced
• Robustness to PVT / PDN
• Post-mitigation $|\rho|$, t

Figure 3.1.: Layer-wise leakage extraction in CiM framework

2. **Array/Architecture** (**A**): Wordline/bitline aggregation, parasitics, and PDN shaping. When hundreds of cells evaluate simultaneously, their individual currents combine along resistive and capacitive wordline and bitline networks, producing spatially coupled and temporally dispersed aggregate signals. Line resistance introduces IR drop that modifies the effective read voltage, while capacitance generates RC time constants that elongate or attenuate the analog response. Shared drivers, sense amplifiers, and DAC elements contribute additional load that shapes the waveform before it reaches the supply. The power-delivery network further filters these currents, embedding package inductance, decoupling capacitance, and rail impedance into the observable signature. This architectural stage, therefore, dictates how localized device events manifest as global supply-level leakage.

3. **Design** (**C**): Compute/sense primitives (e.g., scouting-logic XOR, MAC+ tanh, multi-layer pipelines, CAM-HDC). Different compute primitives exercise the array in distinct ways, creating unique temporal windows where leakage is strongest. XOR-based scouting logic produces short compare-driven spikes,

MAC primitives generate continuous current accumulation, and CAM-style match-line discharge yields exponential decays tied to Hamming distance. Activation functions, comparator thresholds, and ADCs sampling add nonlinearities that modulate the amplitude and slope of the resulting waveform. Multi-layer analog networks further superpose these patterns across depth, creating multi-modal leakage signatures. These design-specific behaviors define the features exploited by profiling, template attacks, correlation analysis, and pattern-based inference in the remainder of the chapter.

3.2.1. Layer D - Device-Level Leakage Models

ReRAM (1T1R). A 1T1R bitcell stores conductance states $G_{\mathrm{LRS}} = 1/R_{\mathrm{LRS}}$ and $G_{\mathrm{HRS}} = 1/R_{\mathrm{HRS}}$. Under a small read bias V_r (linear regime), the cell read current is

$$I_{\mathrm{cell}}(b,t) = \frac{V_r}{R_t + R(b,t)}, \qquad R(b,t) = \bar{R}(b)\big[1 + \delta^{\mathrm{D2D}}\big] + \eta^{\mathrm{RTN}}(t), \tag{3.3}$$

where $b \in \{\mathrm{HRS}, \mathrm{LRS}\}$, R_t is the access/parasitic resistance, δ^{D2D} models device to device variation, and $\eta^{\mathrm{RTN}}(t)$ captures random telegraph noise. Read disturb is negligible for sufficiently small V_r and integration time [22], [63]. In addition, the filamentary conduction mechanism produces strong cycle-to-cycle variability, which introduces low-frequency drift in $R(b,t)$ across repeated reads. The nonlinear voltage dependence of the conductive filament creates small but systematic changes in read current under different stimulus amplitudes, contributing to data correlated analog leakage. The surrounding access transistor also interacts with the filament resistance, amplifying mismatch and temperature sensitivity, and these effects propagate upward during current summation at the array level.

STT-MRAM (1T1MTJ). For an MTJ read at V_r,

$$I_{\mathrm{cell}}(m) = \frac{V_r}{R_t + R(m)}, \qquad R(m) \in \{R_P, R_{AP}\}, \quad \mathrm{TMR} = \frac{R_{AP} - R_P}{R_P}. \tag{3.4}$$

Thermal and $1/f$ noise contribute to read variability; proper biasing avoids read disturb and state flips during inference [42]. Furthermore, voltage-controlled magnetic anisotropy slightly modulates the MTJ resistance under different bias conditions, creating subtle variations in the observed current. The stochastic nature of magnetization in the free layer leads to read time fluctuations that can be exploited as temporal leakage markers. Line resistance and transistor threshold variation introduce additional spread between nominal R_P and R_{AP}, which extends the distinguishability of MTJ states in integrated supply measurements.

Device-layer takeaway. Both technologies decompose naturally into (i) a *deterministic* conductance term set by state and access path, and (ii) a *stochastic* fluctuation term (RTN, thermal variation, device mismatch). These terms are the primitive leakage sources that propagate upward through array summation and PDN filtering. The deterministic component encodes the weight information directly, while the stochastic component provides a source of exploitable temporal diversity across traces. Because these fluctuations remain correlated with operating conditions and stimulus patterns, they generate consistent side channel features after PDN shaping. Understanding this decomposition is essential for constructing accurate leakage predictors and for explaining the trace structures analyzed in subsequent sections.

Consider an activated column j with selected rows $\mathcal{R}$. The bitline current is the superposition of row-weighted cell currents plus parasitics:

$$I_{BL,j}(t) = \sum_{i \in \mathcal{R}} V_{\mathrm{WL},i}(t)\, G_{ij}(t) \; + \; I_j^{\mathrm{leak}}(t) \; + \; \varepsilon_j^{\mathrm{sneak}}(t), \tag{3.5}$$

where $V_{\mathrm{WL},i}(t)$ is the (possibly amplitude-coded) row voltage, $G_{ij}(t)$ follows (3.3) (or its MTJ analog), I_j^{leak} captures line or selector leakage, and $\varepsilon_j^{\mathrm{sneak}}$ lumps residual sneak or coupling effects (suppressed in 1T1X but non-zero with finite R_t). Wordline and bitline resistances produce IR drop and RC dispersion, so $I_{BL,j}(t)$

is temporally shaped even for step inputs. In practical crossbars, these parasitics create column dependent attenuation that modifies the apparent strength of each contributing conductance. The dynamic response also depends on driver strength and rise-time limits, which distort the waveform observed at the supply. Spatial proximity couples neighboring bitlines through capacitance, introducing correlated fluctuations that persist after PDN filtering. These array-level distortions account for much of the structured leakage that attackers later exploit by aligning traces to evaluate windows.

The array and peripheral currents (drivers, DACs or ADCs, sense amplifiers) draw from the supply and are filtered by the PDN:

$$I_{\mathrm{VDD}}(t) = \Big([\,\textstyle\sum_j I_{BL,j}(t)] + I_{\mathrm{periph}}(t) \Big) * h_{\mathrm{PDN}}(t) \;+\; n_{\mathrm{meas}}(t), \tag{3.6}$$

consistent with the readout model in §3.1. In practice, the PDN convolution introduces low-pass characteristics that emphasize slower current components and suppress high-frequency switching noise. Package inductance generates small overshoot and ringing that provide reproducible alignment points across traces. The periphery contributes its own characteristic current bursts, for example ADC sampling edges or sense-amplifier firing, which form reliable temporal landmarks. The combination of array summation and PDN filtering therefore produces a structured current signature that directly reflects the weighted conductance pattern being exercised. This structure becomes the basis of the correlation and template-based predictors used later in the chapter.

An RC surrogate for BL dynamics. A convenient predictor for how input-weight interactions shape energy on column j over an evaluate window t_e uses the bitline time constant

$$\tau_j(x,w) = \frac{C_{\mathrm{BL}}}{G_{\mathrm{act},j}(x,w) + G_{\mathrm{leak}}}, \qquad G_{\mathrm{act},j}(x,w) = \sum_{i\in\mathcal{R}} \alpha_i(x)\, G_{ij},$$

where C_{BL} is the bitline capacitance, G_{leak} is a residual conductance to ground or rails, and $\alpha_i(x) \in [0,1]$ encodes the (possibly analog) row activation implied by input x. For a step-like row drive of amplitude V_r,

$$V_{BL,j}(t_e) = V_r\Big(1 - e^{-t_e/\tau_j}\Big), \qquad E_{\mathrm{BL},j}(x,w) \;\propto\; C_{\mathrm{BL}}\, V_{BL,j}^2(t_e),$$

which is equivalent to the compact form

$$E_{\mathrm{BL},j}(x,w) \propto \left[1 - \exp\!\Big(- \frac{t_e\,[G_{\mathrm{act},j}(x,w) + G_{\mathrm{leak}}]}{C_{\mathrm{BL}}}\Big)\right]^2. \tag{3.7}$$

This surrogate recurs across primitives because it abstracts the same physical mechanism: input-weighted conductance accelerates BL charging, which modulates both instantaneous and integrated supply current after PDN filtering. The model also explains why leakage strength increases when t_e is tuned near the midpoint of the exponential rise, where small changes in $G_{\mathrm{act},j}$ yield large differences in $V_{BL,j}(t_e)$. Variability in C_{BL} and G_{leak} shifts this sensitivity point across columns, producing characteristic asymmetries that tracing algorithms can detect. When multiple columns evaluate concurrently, their time constants become partially correlated through the shared PDN, amplifying column-to-column structure in the measured trace. These properties make the RC surrogate particularly effective for constructing analog-aware leakage predictors in later sections.

3.3. Layer C - Circuit Design-Specific Leakage Realizations

The D/A layer physics manifest differently per primitive. We analyze four representative cases [96], [104], [103].

3.3.1. Case I - Scouting-Logic XOR

Design Motivation

Scouting logic performs Boolean functions in sensing using the analog sum of two cells during a read, preserving states (no write-back). XOR is especially informative for weight recovery and cryptographic evaluation because the decision depends on input or weight mismatch. This enables lightweight "probing" of the stored conductances without modifying them, making it suitable for adversarial interrogation. The primitive also avoids write disturbances and therefore supports repeated queries under identical conditions, which enhances stationarity and reduces averaging noise. Since XOR collapses the two-device state into three distinct current regions, it provides clear analog separation that survives PDN filtering. These properties make scouting-XOR a convenient bridge between continuous analog behavior and downstream digital decision logic.

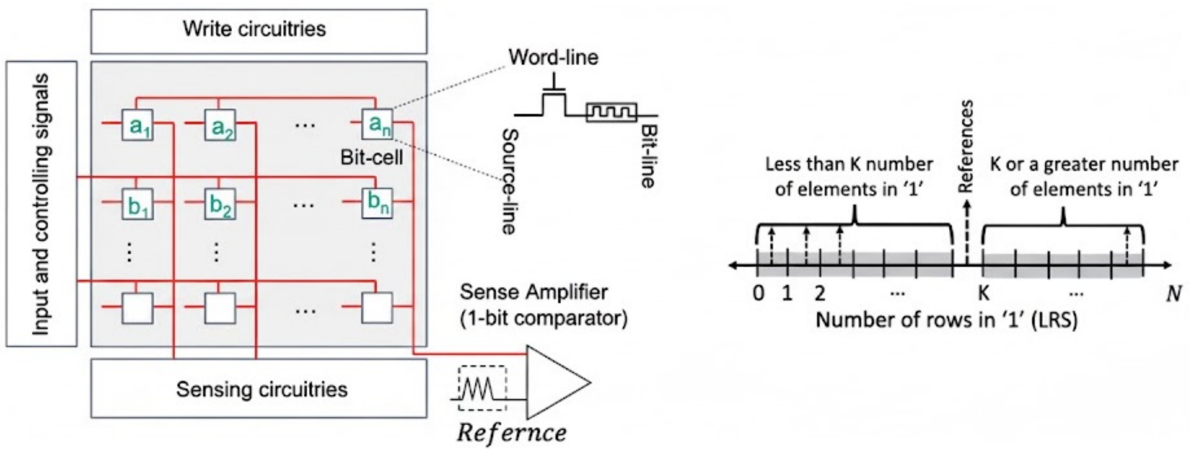


Figure 3.2.: Schematic of a scouting-based logic: different references create different logic operations (XOR, AND, etc.).

Principle of Operation

Each XOR pair uses a weight cell ($w \in \{0, 1\}$) and an input cell ($x_k \in \{0, 1\}$) read simultaneously at V_r. The total conductance is

$$G_{\text{tot}}(x_k, w) = \frac{1}{R_t + R(x_k)} + \frac{1}{R_t + R(w)}. \tag{3.8}$$

With two device states, three current bands (HH), (HL/LH), (LL) map to $\{I_{\text{low}}, I_{\text{mid}}, I_{\text{high}}\}$. A window comparator with references $I_L < I_H$ outputs

$$\text{XOR}(x_k, w) = \begin{cases} 1, & I_L < I_{\text{sense}}(x_k, w) < I_H, \\ 0, & \text{otherwise.} \end{cases} \tag{3.9}$$

This sensing stage introduces analog variations from comparator offset, bias current, and reference ladder mismatch, which slightly distort the boundaries between regions but remain consistent enough for correlation analysis. The pairwise read operation also emphasizes mismatch-driven behavior, making the mid-band particularly sensitive to device noise and parasitics. These variations amplify distinguishability between hypotheses, improving predictor performance during CPA.

Device/Array to Leakage

During evaluate, BL discharges via $R_{\text{eq}} = 1/G_{\text{tot}}$, producing

$$I_{\text{BL}}(t) \simeq V_r\, G_{\text{tot}}(x_k, w)\, (1 - e^{-t/\tau_{\text{BL}}}), \qquad \tau_{\text{BL}} = \frac{C_{\text{BL}}}{G_{\text{tot}} + G_{\text{leak}}}. \tag{3.10}$$

Supply current adds comparator toggling filtered by the PDN, yielding three observable bands. The RC trajectory also shifts depending on line resistance and driver rise-time, introducing column-specific distortions that persist across repeated evaluations. When the comparator fires, its regenerative action generates a sharp current surge that produces a high-SNR peak after PDN filtering. Subtle RTN or thermal variations in either cell modulate the early part of the discharge curve, making the evaluate window particularly useful for analog leakage extraction. These combined dynamics create a multi-window signature where both smooth and impulsive features encode the secret w.

Leakage Predictors

RC discharge (evaluate window):

$$\hat{p}_k^{\mathrm{RC}} = \left[1 - \exp\left(-\frac{t_e\,[G_{\mathrm{tot}}(x_k, w) + G_{\mathrm{leak}}]}{C_{\mathrm{BL}}}\right)\right]^2. \tag{3.11}$$

Comparator activity (sense window):

$$\hat{p}_k^{\mathrm{XOR}} = x_k \oplus w. \tag{3.12}$$

Shaping with window templates $g_{\mathrm{eval}}(t)$, $g_{\mathrm{SA}}(t)$ gives

$$\hat{\mathbf{p}}_k(t) = \alpha\,\hat{p}_k^{\mathrm{RC}}\,g_{\mathrm{eval}}(t) + \beta\,\hat{p}_k^{\mathrm{XOR}}\,g_{\mathrm{SA}}(t). \tag{3.13}$$

The templates approximate the characteristic waveform envelopes produced by BL discharge and SA regeneration, allowing predictors to remain effective even under heavy PDN filtering. The additive combination captures the fact that evaluate and sense windows are temporally disjoint and map to different physical origins. In practice, the evaluate predictor dominates under strong decoupling, while the comparator predictor becomes stronger when the PDN emphasizes high-frequency content. Weight estimation, therefore, benefits from scanning both windows to locate the dominant leakage mode.

End-to-End Framework

Figure 3.3 summarizes the simulation-driven methodology used in this chapter to analyze the CPA vulnerability of different CiM implementations. The workflow begins by instantiating device-level process variation for each chip instance, using either resistive distributions for ReRAM-based CiM or transistor and bitcell-width variations for SRAM-based baselines. These variations fix the physical conductance or delay profile of the instantiated array and create the stationary leakage characteristics that an adversary later exploits. After device sampling, the framework generates power traces for the full input space by applying all input patterns while keeping the target weight fixed, mimicking the attacker's ability to repeatedly probe the same stationary secret.

The generated raw currents are then passed through an electrical-level SPICE simulation that includes realistic wordline and bitline parasitics, access-transistor responses, and sense-path dynamics. At this stage, we insert the PDN model, as shown in Figure 3.4, with package-level and on-chip RLC parameters, ensuring that the observable current waveform reflects the same filtering and distortion seen in a real system. The PDN convolution preserves the input-weight correlation but attenuates and temporally disperses the signature, making the attack more representative of physical measurements. A measurement-noise model is then applied to emulate oscilloscope or shunt-based probing noise, which allows controlled experimentation across different SNR conditions.

Once traces are collected, they are processed by the CPA engine, which computes correlation between predicted leakage features and the measured waveforms. The framework cleanly separates the roles of physical modeling and statistical evaluation, allowing the same CPA engine to test multiple CiM primitives under identical attacker assumptions. Because all leakage predictors are derived from the earlier D/A/C layers, the same pipeline supports scouting-XOR, analog MAC, multi-layer CiM, and CAM-HDC cases. This modularity also enables systematic sensitivity analysis with respect to PDN strength, device noise, array dimensions, and input ensembles. The overall methodology provides a controlled environment for quantifying trace-to-disclosure, leakage windows, and the comparative robustness of different technologies before validating insights on real hardware in later chapters.

As setup, we generate power traces by performing circuit-level simulations in the Cadence Virtuoso simulator. The CiM architecture with NVM and SRAM in a 257-row crossbar array is implemented using a $22nm$ Global Foundries technology. We perform a Monte Carlo simulation for each state (LRS and HRS) of the NVM with 1000 samples fixed at room temperature to get the resistance distribution. The device models of the NVM used in the simulation and details about the simulation setup are enlisted in Table 3.2. The values of all R, L, and C for PDN are suggested from [72], while for the capacitor we have used C_{die}=320 nF, scaled to the maximum expected total current in our circuits for all technologies. Additional MN is later injected computationally into the simulated traces using Python script. One million traces were generated for each of the masking and hiding designs.

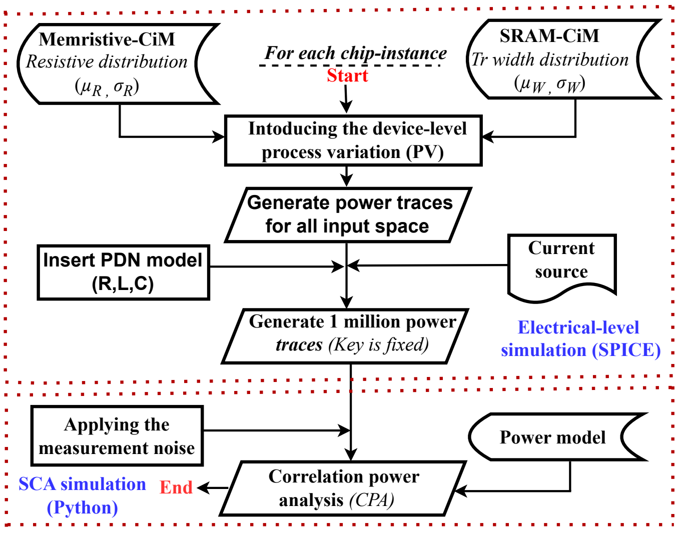

Memristive-CiM
Resistive distribution
(μ_R, σ_R)
For each chip-instance
Start
SRAM-CiM
Tr width distribution
(μ_W, σ_W)
Intoducing the device-level process variation (PV)
Generate power traces for all input space
Insert PDN model (R,L,C)
Current source
Generate 1 million power traces (Key is fixed)
Electrical-level simulation (SPICE)
Applying the measurement noise
Power model
SCA simulation (Python)
End
Correlation power analysis (CPA)

Figure 3.3.: End-to-end simulation framework for evaluating the CPA vulnerability of CiM operations. The pipeline incorporates device-level variation, electrical-level SPICE simulation, PDN filtering, noise injection, and statistical CPA analysis.

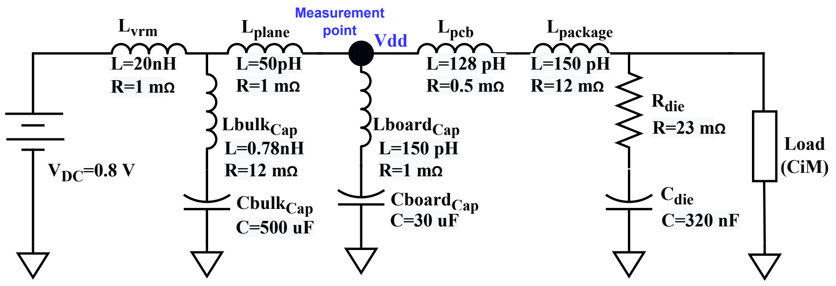


Figure 3.4.: End-to-end simulation framework for evaluating the CPA vulnerability of CiM operations. The pipeline incorporates device-level variation, electrical-level SPICE simulation, PDN filtering, noise injection, and statistical CPA analysis.

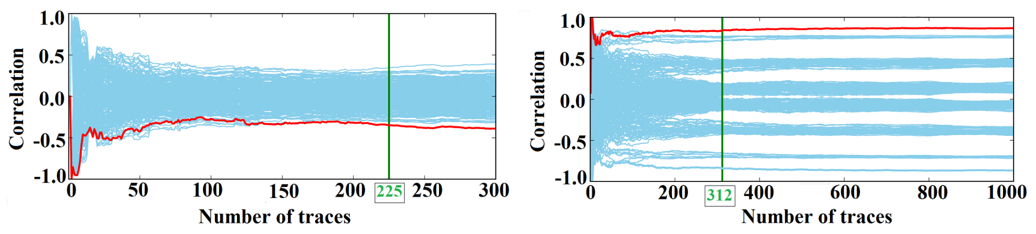
CPA Workflow

Given trace $T_k(t)$ and hypothesis $w_h \in \{0, 1\}$:

1. Compute $G_{\text{tot}}(x_k, w_h)$ and $\hat{p}_k^{\text{RC}}$ via (3.11),
2. Compute $\hat{p}_k^{\text{XOR}} = x_k \oplus w_h$,
3. Project trace onto $g_{\text{eval}}(t)$, $g_{\text{SA}}(t)$ to get $y_k^{\text{eval}}, y_k^{\text{SA}}$,

Table 3.2.: Simulation parameters

MTJ model [39]	-Radius = 20 nm - RA = $7.5\Omega\mu m^2$ - Nominal TMR = 150%
VCM-based device model: *JART VCM v1b Read variability* [80]	- Filament radius = 45 nm - Length of the disc region = 0.6 nm - Initial oxygen vacancies concentration in the disc $[10^{26}/m^3]$ for 'LRS' = 3, for 'HRS' = 0.009
Temperature	$27°C$



(a) **STT-MRAM**-based CiM **(b)** **SRAM**-based CiM

Figure 3.5.: CPA on different CiM implementations of an 8-bit XOR operation with the minimum number of traces for key recovery (marked green) in all plots. For all attacks, the effects of the PDN and MN are considered.

4. Evaluate Pearson correlation:

$$\rho(\mathbf{a}, \mathbf{b}) = \frac{\mathrm{Cov}(\mathbf{a}, \mathbf{b})}{\sqrt{\mathrm{Var}(\mathbf{a})\,\mathrm{Var}(\mathbf{b})}}, \tag{3.14}$$

i.e., $\rho_{\text{eval}} = \mathrm{corr}(y^{\text{eval}}, \hat{p}^{\text{RC}})$, $\rho_{\text{SA}} = \mathrm{corr}(y^{\text{SA}}, \hat{p}^{\text{XOR}})$,

5. Pick w_h maximizing $|\rho_{\text{eval}}|$, $|\rho_{\text{SA}}|$ (or window-wise R^2).

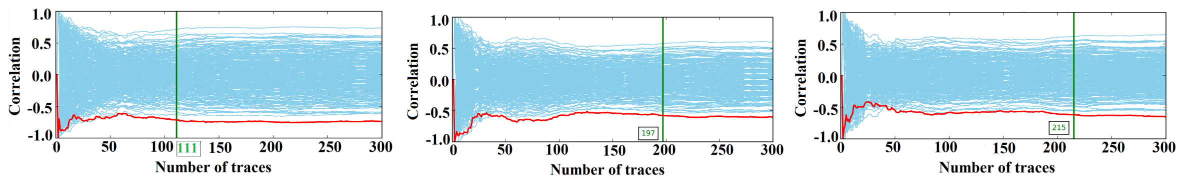
Trace-to-disclosure: $N_{\text{TD}} \approx z^2/\rho_\star^2$, with dominant $\rho_\star$. The dual-window structure offers robustness because inaccuracies in one window can be compensated by the other. Projection step effectively isolates leakage-rich components, reducing noise sensitivity and improving alignment tolerance. The hypothesis scan is cheap due to binary w, enabling rapid profiling of all weights in parallel. The workflow also scales naturally to multi-bit cells by extending the hypothesis space, a capability used in later CiM attack scenarios.

Table 3.3.: Leakage predictors and timing windows for scouting-logic XOR.

Window	Predictor	Physical meaning	Target secret
Evaluate	$\hat{p}^{\text{RC}}$	BL discharge energy $\propto G_{\text{tot}}$	w
Sense (SA)	$\hat{p}^{\text{XOR}}$	Comparator toggling on mismatch	w
Precharge	HD(driver)	WL/BL driver activity (weak)	–

3.3.1.1. CPA-based Vulnerability Assessment of Unprotected CiM Designs

The plots in this section correspond to the progress of correlation coefficients for different key bytes at a particular time stamp with respect to the number of traces. All of these results are reported on an exemplary chip instance.



(a) CPA without any noise **(b)** CPA with the effects of the PDN **(c)** CPA with the effects of PDN and MN

Figure 3.6.: CPAs on ReRAM-MRAM-based CiM. Effect of adding the PDN and MN to the simulation flow on the vulnerability of the design, shown by the amount of measurements needed for key recovery (marked green).

Table 3.4.: Minimum number of traces needed to attack various CiM technologies, based on simulations including PV, PDN and MN. The result is averaged over 10 different chip instances.

CiM Technology	Traces needed by simulation type		
	PV	PV+PDN	PV+PDN+MN
Phase-Change Memory (PCM)	21	41	47 ± 4
ReRAM	111	197	211 ± 15
STT-MRAM	143	189	228 ± 26
SRAM	212	276	322 ± 53

The red lines correspond to the correlation values with the correct key byte, whereas the blue lines correspond to the correlation with incorrect keys. With green vertical lines, we denote the number of traces, where key recovery is successful. As expected, when using a Hamming weight power model on an XOR operation, all of the plots are symmetric, apart from noise effects. This also means that for each key byte both the correct key as well as the bitwise complement show the highest absolute correlation values, which can however easily be brute-forced by an attacker for a full key recovery.

We show how the correlation coefficients are changing with different amounts of traces in the case of ReRAM in Figure 3.6c. For the remaining CiM implementations, we present the results of performing CPA attacks in Figure 3.5.

As we apply the CPA on the power traces generated in the simulation environment, we also analyze the effect of PDN noise and measurement noise on the vulnerability. We observe that both effects increase the amount of traces required for key recovery to some extent. However, they all are still vulnerable with more traces. Exemplary, we show the increase in the minimum number of traces for a successful attack in ReRAM-based CiM in Figure 3.6.

These results were all performed on one chip instance. As one instance might be particularly difficult or easy to attack (depending on PV), we cannot generalize these results. Thus, we additionally simulated 9 more chip instances to cross-check our results. We report our summarized results with the average amount of traces needed as well as their variances across all chip instances in Table 3.4.

3.3.1.2. Proposed CiM-friendly Masking Protection

Block ciphers usually allow implementing the non-linear operations like S-Box as look-up tables, which are included in our CiM design to account for a realistic implementation. We then apply masking on the S-Boxes to strengthen the cryptographic module against side channel attacks, by generating masked look-up tables T_m with the following property $T_m(v \oplus m) = T(v) \oplus m$. That means, for each used mask m, we will generate a separate masked look-up table for the corresponding non-linear operation.

The designer of a cryptographic system may reuse a limited number of masking values as a trade-off between security and the required amount of memory. To compensate for the negative effects on security, the masks can be refreshed to new ones after a certain amount of encryption.

There are several advantages of the described masking scheme with respect to CiM architectures. First, there is no need for any changes to the hardware, as we only need to extend the memory array. The memory overhead increases with the number of masks that are used. Pre-computing the masked S-Box look-up tables is negligible, as it needs to be done only once. Another advantage is that the true random numbers which are needed to randomly select a mask in the CiM can be generated by memristors themselves.

In this work, we reuse 16 masks that correspond to 16 masked S-Box look-up tables in the memory. We choose these masks randomly from the pool of all available mask values, for which we pre-compute the masked SBOX tables in an algorithmic way. The inputs themselves are not masked, and masking is automatically applied within the CiM-based XOR operation for $SBox_m(input) \oplus key_m$ by randomly choosing one of the masked look-up tables in the CiM-enabled memory. For each used mask we also pre-compute a masked key key_m and store it next to the masked look-up table in CiM-enabled memory, so adding the key with CiM-based XOR returns an already unmasked output. This way, the intermediate values that are processed are randomized, and the power consumption becomes independent of the secret data.

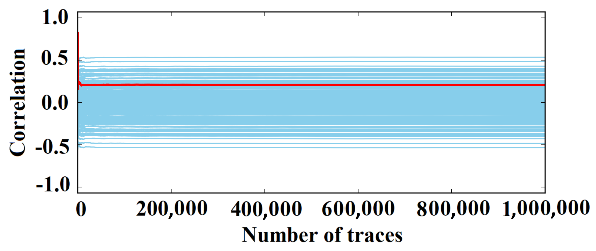
3.3.1.3. Proposed Hiding Protection

We separately implement another type of countermeasures called *hiding*. *Hiding* countermeasures aim to hide circuit activity by reducing the SNR between data-dependent differences in power consumption and power noise.

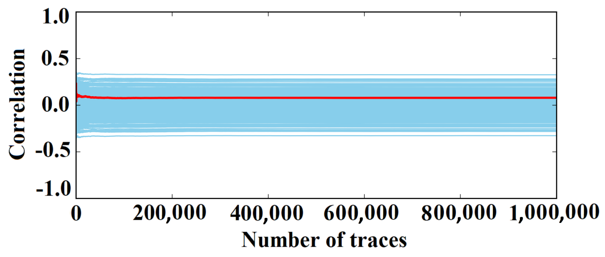
In this paper, we follow the concept of power equalization [25] by adding duplicated logic that shall mimic the inverse behavior of our actual circuit. In CMOS circuits, custom *inverted* circuits have to be designed or dual-rail logic has to be used. For CiM we can benefit from the specific properties of a regular memory array for a hiding solution that can be more easily transferred to other algorithms as well.

In our specific design, we extend the original 257×8 cell array with a complementary 257×8 array, doubling the columns, leading to an overall array size of 257×16. We operate half of the CiM columns as usual, where one row contains the secret key and the other 256 rows contain the AES S-Box look-up. The other half of the array stores a bitwise complementary key and bitwise complementary S-Box look-up. During the CiM XOR operation, the total amount of activated memristor cells in HRS and the total amount of activated cells in LRS is data-independent, namely 16 each.

In an ideal circuit, the resulting power consumption is equal for all input values, whereas our more realistic circuit contains process and design variations that also lead to runtime variations later on. These remaining variations result in side-channel leakage, which will eventually allow key recovery. However, the number of required measurements will be significantly increased, which is demonstrated in our results.



(a) CPA on hiding protected design



(b) CPA on masking protected design

Figure 3.7.: CPA performed on ReRAM-based CiM-design on which our protections are applied, showing in both cases key recovery is unsuccessful.

3.3.2. Case II - MAC + tanh activation Layer

Design Motivation

Analog MAC+tanh is the workhorse of ACIM neural inference: weights are encoded as cross-point conductances, inputs are driven as row voltages, and the resulting column current is integrated and passed through an analog tanh activation. Because the accumulated current is proportional to $\boldsymbol{G}^{\top}\boldsymbol{v}$, its magnitude preserves information about the signed and multi-bit weight encoded in the conductances, leading to amplitude-bearing leakage that is significantly richer than the binary decision of scouting logic. The tanh block further introduces a smooth nonlinearity that compresses large positive or negative currents, creating distinct analog regions (linear, semi-saturated, and fully saturated) that can be exploited by an adversary to infer both magnitude and sign of weights. Low-frequency components of the MAC trajectory are also preserved by the PDN more strongly than high-frequency edges, making this primitive particularly vulnerable under strong on-die decoupling. A block-level schematic of the primitive is shown in Fig. 3.8.

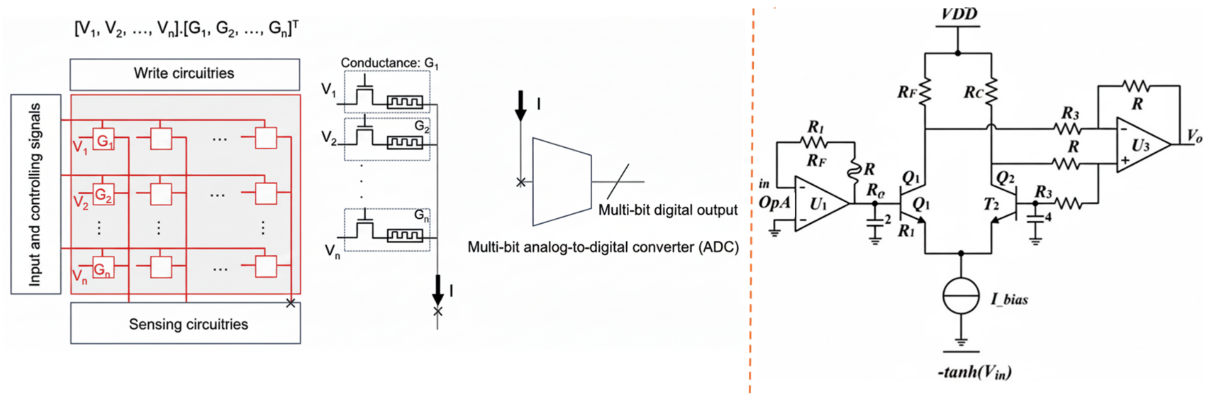


Figure 3.8.: Schematic of a MAC+tanh column: weighted current summation followed by a differential-pair tanh activation and optional quantization.

Principle of Operation

For column j and trial k, the instantaneous MAC current is

$$I_{\mathrm{MAC},j}(k) = \sum_{i=1}^{N_{\mathrm{in}}} V_{i,k}\, G_{ij}, \tag{3.15}$$

which is converted and fed to a differential-pair tanh:

$$V_{\mathrm{out},j}(k) = V_{\mathrm{bias}}\, \tanh\left(\frac{I_{\mathrm{MAC},j}(k)\, R_L}{2 I_T}\right), \tag{3.16}$$

with load R_L and tail current I_T. Two windows dominate the observable supply current: the *MAC evaluate* interval (summation/integration) and the *activation* interval (nonlinear bias modulation around the diff-pair).

Device/Array to Leakage

With many active rows, the effective column conductance seen during evaluate is

$$G_{\mathrm{col},j}(k) = \sum_{i} x_{i,k}\, G_{ij}, \tag{3.17}$$

which sets both the amplitude and slope of $I_{\mathrm{MAC},j}(t)$. This trajectory is convolved with the PDN impulse response, producing a smooth, high-SNR analog signature. Device-level phenomena, such as cell-to-cell variation, RTN, line resistance, and BL/WL driver slew, introduce stable, instance-specific distortions that persist across queries and can be profiled. The activation stage adds a localized burst of current near the operating point where the differential pair exhibits large transconductance (near its linear region). At deeper saturation, the activation contribution flattens and becomes less informative, while the evaluate window remains amplitude-coded.

Leakage Predictors

Evaluate (RC discharge). A compact energy predictor that captures PDN-filtered evaluate leakage is

$$\hat{p}_k^{\mathrm{MAC}} = \left[1 - \exp\left(-\frac{t_e\,[G_{\mathrm{col},j}(k) + G_{\mathrm{leak}}]}{C_{\mathrm{BL}}}\right)\right]^2. \tag{3.18}$$

Activation (nonlinear transconductance). For a differential-pair tanh, the small-signal transconductance scales as $g_m \propto I_T\ \mathrm{sech}^2(\cdot)$. Integrating over the activation window motivates

$$\hat{p}_k^{\tanh} = \mathrm{sech}^2\Big(\gamma\, G_{\mathrm{col},j}(k)\Big), \tag{3.19}$$

with a fitted γ absorbing gains and level shifts.

Shaped time predictor. We shape predictors with window templates $g_{\mathrm{MAC}}(t)$ and $g_{\tanh}(t)$:

$$\hat{\mathrm{p}}_k(t) = \alpha\, \hat{p}_k^{\mathrm{MAC}}\, g_{\mathrm{MAC}}(t) + \beta\, \hat{p}_k^{\tanh}\, g_{\tanh}(t), \tag{3.20}$$

mirroring the disjoint physical origins of evaluate and activation leakage. *For quick lookup, the mapping between windows, predictors, and secrets is summarized in Table 3.5.*

Table 3.5.: Predictors and timing windows for MAC+tanh.

Window	Predictor	Physical meaning	Target secret
MAC Evaluate	$\hat{p}^{\mathrm{MAC}}$	BL discharge $\propto \sum_i x_i G_i$	**W**
Activation	$\hat{p}^{\tanh}$	$g_m \propto \mathrm{sech}^2(\cdot)$	**W**
Precharge	HD(driver)	Driver toggling (weak)	–

CPA/Hypothesis Testing

For hypothesis weights $\mathbf{W}_h$ and trial k,

$$\hat{p}_k^{(1)} = \sum_i x_{i,k}\, G_{ij}(w_{ij,h}), \tag{3.21}$$

$$\hat{p}_k^{(2)} = \mathrm{sech}^2(\gamma\, \hat{p}_k^{(1)}). \tag{3.22}$$

Project the measured trace $T_k(t)$ onto $g_{\mathrm{MAC}}(t)$ and $g_{\tanh}(t)$ to obtain y_k^{MAC} and $y_k^{\tanh}$. Evaluate Pearson correlations $\rho_{\mathrm{MAC}} = \mathrm{corr}(y^{\mathrm{MAC}}, \hat{p}^{(1)})$ and $\rho_{\tanh} = \mathrm{corr}(y^{\tanh}, \hat{p}^{(2)})$, then select the hypothesis maximizing $|\rho_{\mathrm{MAC}}|+|\rho_{\tanh}|$. The traces-to-disclosure heuristic is $N_{\mathrm{TD}} \approx z^2/\rho_\star^2$ with dominant $\rho_\star$.

Vulnerability Analysis

Figures 3.9 and 3.10 illustrate leakage features around activation edges and template separability. In our unprotected design with 16 simultaneous row activations, both CPA and TVLA confirm exploitable leakage from the ADC/quantizer stage (see Figure 3.11): the correlation peak across candidate weights grows with trace count, and the TVLA curve exceeds the $|t|$>4.5 threshold at the same windows where the ADC toggles. Across technologies and PDN strengths, the quantitative trends for the evaluate/activation windows are consolidated in Table 3.6, while the impact of ADC resolution and dithering is summarized in Table 3.7.

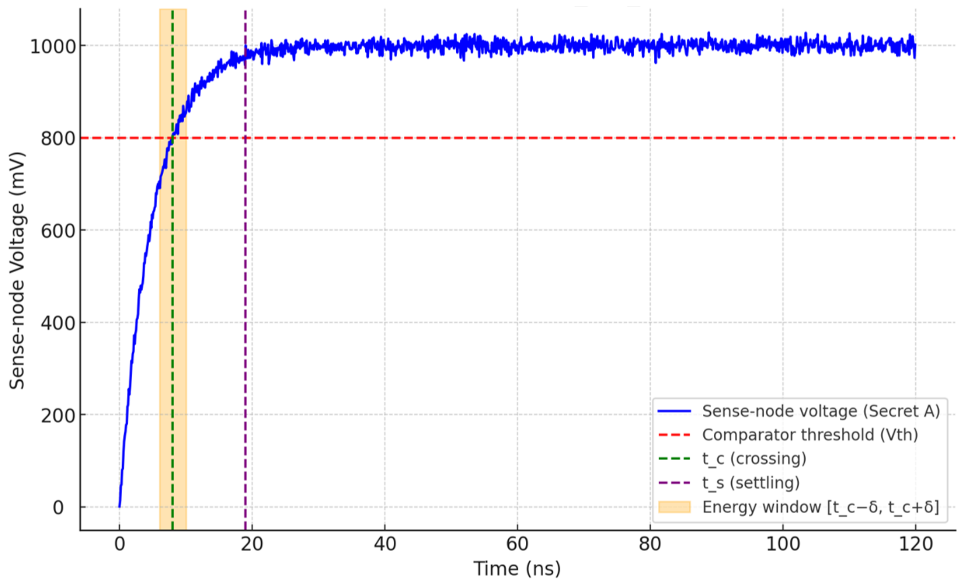


Figure 3.9.: Feature extraction around evaluate and activation: timing landmarks, local energy, slope at the activation edge, and comparator fault flag.

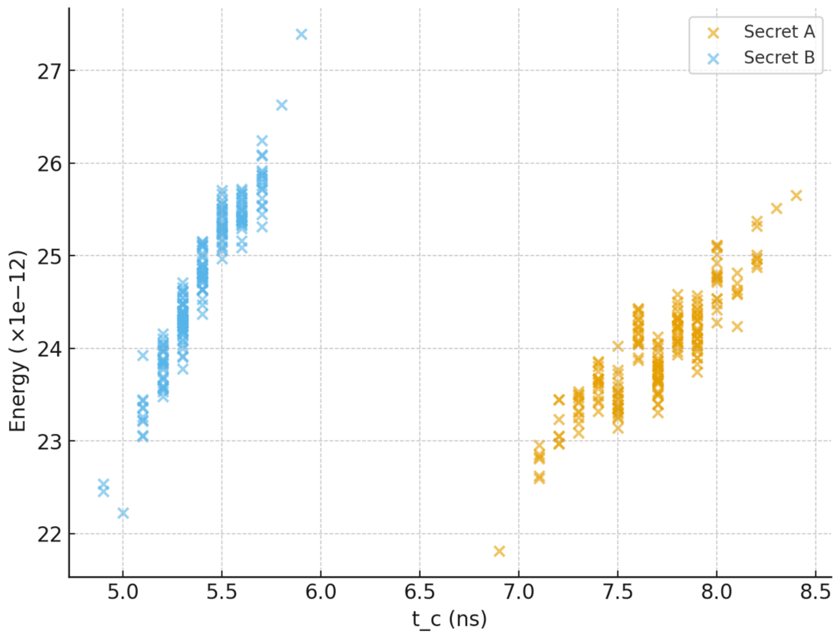
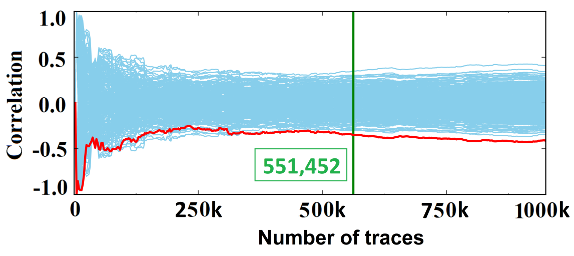


Figure 3.10.: Template separation for different weight codes using activation-edge features; clusters remain linearly separable even under PDN filtering.

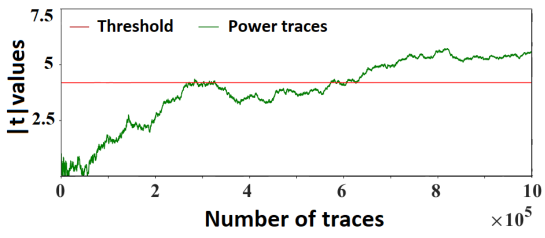
Countermeasure: Balanced Comparator/Dummy Path

To reduce activation-stage leakage, we instantiate a duplicate, cross-biased comparator that draws complementary current, equalizing power across input common-mode variations (concept in Figure 3.12). The measured/simulated power versus input CM voltage becomes nearly flat after protection (Figure 3.13), and both CPA and TVLA attacks fail on the protected sub-array (Figure 3.14).

Practical Notes. MAC leakage enables multi-bit recovery; activation produces short bursts near zero crossings. ReRAM provides larger dynamic range but more RTN; STT-MRAM is quieter but smaller in amplitude. PDN decoupling trades evaluate-window correlation (stronger with more decap) against activation-edge visibility (weaker with more decap). Comparator offset and metastability add timing jitter



(a) CPA on ADC outputs



(b) TVLA on ADC windows

Figure 3.11.: CPA and TVLA on an unprotected MAC+tanh path (16 rows active). Minimum trace counts for disclosure are marked in green in the original plots.

Table 3.6.: MAC+tanh leakage as a function of technology and PDN decoupling. Peak Pearson correlation $|\rho|$ is reported for the *evaluate* window (RC discharge) and the *activation* window (tanh bias modulation). N_{TD} is the traces-to-disclosure estimate using $N_{TD} \approx z^2/\rho^2$ with $z = 5$.

Technology	PDN decap	Evaluate window		Activation window	
		$\lvert\rho\rvert$	N_{TD}	$\lvert\rho\rvert$	N_{TD}
ReRAM	Low	0.21	567	0.14	1276
MRAM	Low	0.18	772	0.11	2067
ReRAM	Med.	0.24	435	0.16	977
MRAM	Med.	0.20	625	0.13	1480
ReRAM	High	0.27	343	0.18	772
MRAM	High	0.23	473	0.15	1112

that can be exploited by template attacks but weakens raw CPA, reinforcing the benefit of feature-based profiling.

3.3.3. Case III - Multi-Layer MAC + tanh activation Network

Motivation and Overview

Deep analog networks cascade MAC+tanh blocks. A shared PDN superposes per-layer currents so the observed supply current is a sum of filtered layer activities:

$$I_{\mathrm{VDD}}(t) = \sum_{\ell=1}^{L} \left[I_{\mathrm{array}}^{(\ell)}(t) * h_{\mathrm{PDN}}(t) \right]. \tag{3.23}$$

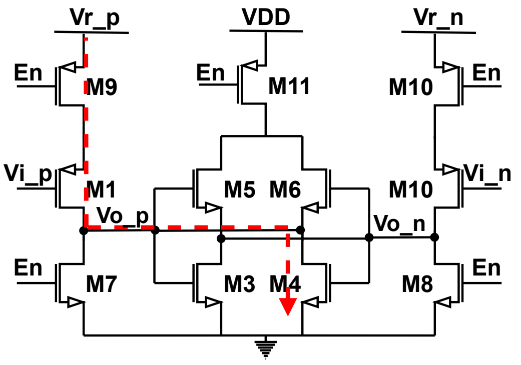
For layer ℓ: $I_{\mathrm{MAC},j}^{(\ell)}(k) = \sum_i V_{i,k}^{(\ell)} G_{ij}^{(\ell)}$ and $V_{i,k}^{(\ell+1)} = \tanh(\beta_\ell\, I_{\mathrm{MAC},i}^{(\ell)}(k))$. Early layers operate in a wider linear region and therefore leak stronger amplitude information; deeper layers are progressively compressed by the preceding tanh stages and contribute shorter, smaller features.

Device/Array to Leakage

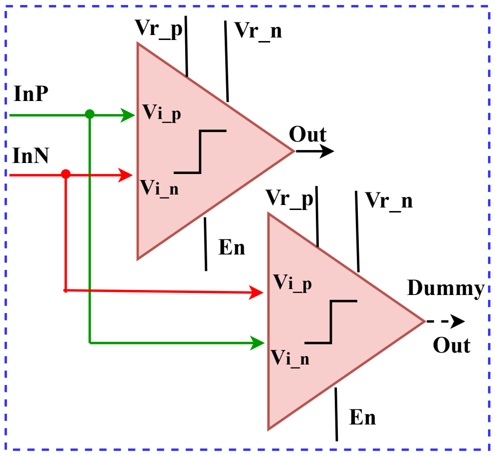
The PDN acts as a linear mixer, coupling layers temporally and spectrally. Unequal evaluate durations, duty cycles, and layer-local decoupling produce distinct “leakage colors” that can be separated by time-frequency weighting. Instance-specific offsets (e.g., SA thresholds, line RCs) persist across trials and can be profiled, while device noise (RTN/thermal) adds colored, partially correlated perturbations.

Table 3.7.: Impact of ADC resolution and dithering on MAC+tanh CPA. Median traces to disclosure (N_{TD}) and empirical success rate after 1000 traces, averaged over several seeds and PDN states. Dithering adds small, uniformly distributed input noise at the ADC to break deterministic binning.

Configuration	Median N_{TD}	Success @ 1000 traces
6-bit ADC (no dither)	360	100%
8-bit ADC (no dither)	430	97%
10-bit ADC (no dither)	540	90%
8-bit ADC + dither	690	81%



(a) Dynamic latch comparator



(b) Balanced comparator with dummy path

Figure 3.12.: Unprotected and protected comparators used in the activation/quantization stage.

Predictors and Shaping per Layer

We extend (3.18)-(3.19) layer-wise:

$$\hat{p}_k^{\mathrm{MAC},(\ell)} = \left[1 - \exp\left(-\frac{t_e^{(\ell)}\,[G_{\mathrm{col},j}^{(\ell)}(k) + G_{\mathrm{leak}}]}{C_{\mathrm{BL}}^{(\ell)}}\right)\right]^2, \quad \hat{p}_k^{\mathrm{tanh},(\ell)} = \mathrm{sech}^2\left(\gamma_\ell\, G_{\mathrm{col},j}^{(\ell)}(k)\right), \tag{3.24}$$

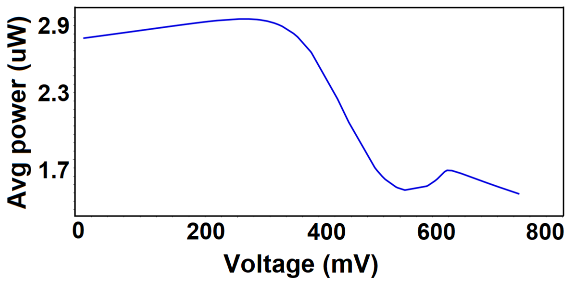
and construct

$$\hat{\mathbf{p}}_k^{(\ell)}(t) = \alpha_\ell\, \hat{p}_k^{\mathrm{MAC},(\ell)}\, g_{\mathrm{MAC},\ell}(t) + \beta_\ell\, \hat{p}_k^{\mathrm{tanh},(\ell)}\, g_{\mathrm{tanh},\ell}(t). \tag{3.25}$$

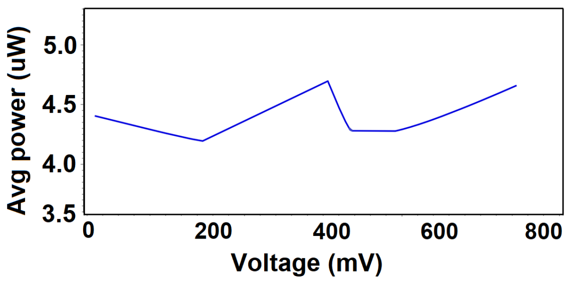
Monotonic tanh lets correlation propagate: $\hat{p}_k^{\mathrm{MAC},(\ell+1)} = f(\mathbf{W}^{(\ell+1)}, \tanh(\mathbf{W}^{(\ell)}\mathbf{x}_k))$, albeit with attenuation. *Numerical results for a three-layer ReRAM MLP are provided in Table 3.8 and visualized in Fig. 3.15.*

Profiling Attack: Layer-wise PCA + Likelihood

Classical CPA deteriorates under layer superposition and compression. We therefore adopt a profiling pipeline that (i) windows traces per layer using coarse timing markers, (ii) applies PCA inside each window to suppress uncorrelated noise and PDN spillover, and (iii) models the profiled PC vectors with a simple Gaussian likelihood per hypothesis. For a victim, we select the hypothesis maximizing the per-layer likelihood and fuse layers by a weighted sum of log-likelihoods. Pre-processing (baseline removal, mild band-selection, alignment, and per-window z-score) follows the common steps detailed in §3.3.5.

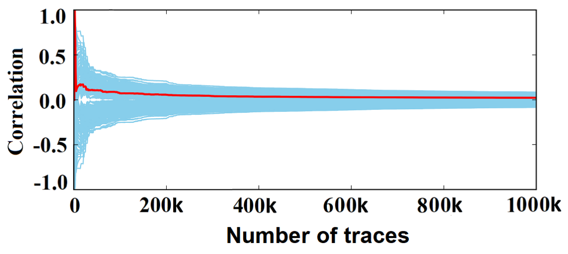


(a) Unprotected: unbalanced power

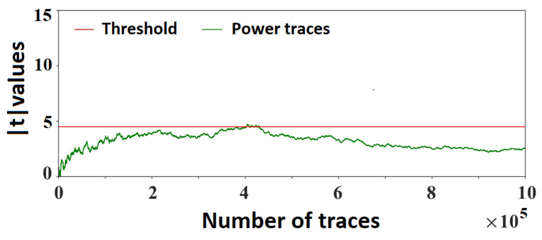


(b) Protected: balanced power

Figure 3.13.: Comparator power vs. input common-mode voltage before and after protection.



(a) CPA fails on protected array



(b) No TVLA leakage after protection

Figure 3.14.: Failed CPA and TVLA on the protected CiM sub-array (8 rows active).

Algorithm 1 Layer-wise PCA + Likelihood for Multi-Layer ACIM

```
Require: Profiling traces $P^{(\ell)}(t)$ per layer $\ell$
Ensure: Recovered $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}$
1: for $\ell = 1$ to $L$ do
2:     Window $\rightarrow P^{(\ell)}(t)$; compute PCA projection $W_{\mathrm{PCA},\ell}$ and retain top PCs ($\geq 90\%$ variance)
3:     Fit Gaussian $p(\mathbf{L}^{(\ell)}|K)$ on PC vectors for weight/class hypothesis $K$
4:     For victim, compute $\mathcal{L}^{(\ell)}(K) = p(\mathbf{L}^{(\ell)}_{\mathrm{victim}}|K)$
5: Fuse: $\log \mathcal{L}_{\mathrm{tot}}(K) = \sum_\ell \omega_\ell \log \mathcal{L}^{(\ell)}(K)$; output $K^* = \arg\max_K \log \mathcal{L}_{\mathrm{tot}}(K)$
```

Decision Metrics. Beyond $|\rho|$, we report layer-wise likelihoods $\mathcal{L}^{(\ell)}$, Mahalanobis distances between profiled clusters, PCA variance ratios, and TVLA across windows to show leakage presence and identifiability under PDN coupling. Complementary frequency-domain statistics are reported in Table 3.9 and visualized in Fig. 3.16.

Countermeasure: Timing Diversification

To blunt profiling, we introduce small, tunable per-bit delays in the activation/comparator path, randomized across inferences (see Figure 3.17). The resulting time warps decorrelate layer windows across traces while preserving correctness.

Practical Notes. Superposition yields compound peaks; early layers contribute broad, high-energy windows, later layers contribute short, saturated bursts. Shared PDN couples all blocks; decap/gating choices affect every layer simultaneously. PCA is robust to mild misalignment and captures instance-specific structure introduced by PDN poles and sense-path offsets.

Table 3.8.: Layer-wise leakage in a 3-layer analog MLP (ReRAM, medium decoupling). Correlation and traces-to-disclosure are reported for *evaluate* and *activation* windows per layer. N_{TD} uses $z = 5$. The first layer leaks the most, deeper layers are attenuated by the tanh nonlinearity and PDN superposition.

Layer	Evaluate window		Activation window	
	$\lvert\rho\rvert$	N_{TD}	$\lvert\rho\rvert$	N_{TD}
L1	0.22	517	0.15	1112
L2	0.17	866	0.12	1737
L3	0.12	1737	0.09	3087

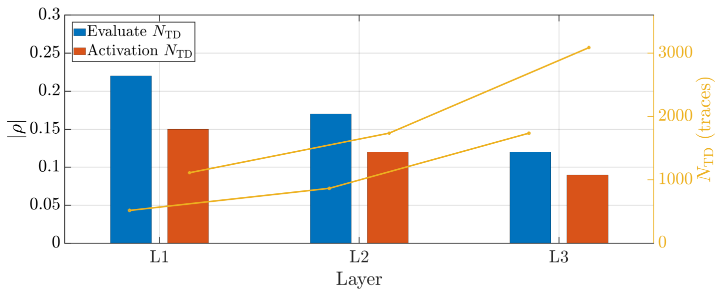


Figure 3.15.: Layer-wise leakage: grouped bars show $|\rho|$ in evaluate/activation windows for L1-L3; lines (right axis) show N_{TD} computed with $z = 5$. L1 dominates; deeper layers are attenuated by tanh and PDN superposition.

3.3.4. Case IV - CAM-Based HDC

Design Motivation and Operation

HDC accelerators commonly use CAMs for similarity search over stored class hypervectors. For row r storing $\mathbf{w}_r$ and query $\mathbf{x}$, match-lines (MLs) are precharged to V_{pre} and then released; each bit mismatch opens a discharge path. The ML current is

$$I_{\text{ML},r}(t) = \sum_{i=1}^{N} V_{\text{read}}(t)\, G_{ri}\, [x_i \oplus w_{ri}], \tag{3.26}$$

thus proportional to Hamming distance (HD). The total supply current sums all rows via the PDN:

$$I_{\text{VDD}}(t) = \sum_r [I_{\text{ML},r}(t) * h_{\text{PDN}}(t)]. \tag{3.27}$$

Match events (small HD) yield slow ML discharge and reduced dynamic current; mismatches (large HD) accelerate discharge and increase current, creating a direct power-HD mapping. The CAM/HDC primitive is sketched in Fig. 3.18.

Predictors and Shaping

With ML capacitance C_{ML} and effective mismatch conductance $G_{\text{eff},r} = \sum_i G_{ri}[x_i \oplus w_{ri}]$, the ML voltage decays as

$$V_{\text{ML},r}(t) = V_{\text{pre}} e^{-t/\tau_r}, \qquad \tau_r = \frac{C_{\text{ML}}}{G_{\text{eff},r}}.$$

We use an RC-energy predictor

$$\hat{p}_r^{\text{RC}} = \left[1 - \exp\left(-\frac{t_e(G_{\text{eff},r} + G_{\text{leak}})}{C_{\text{ML}}}\right)\right]^2, \tag{3.28}$$

and a logical HD predictor

$$\hat{p}_r^{\text{HD}} = \text{HD}(\mathbf{x}, \mathbf{w}_r) = \sum_i [x_i \oplus w_{ri}]. \tag{3.29}$$

Table 3.9.: Frequency-domain separability of layer windows under different PDN decoupling. Δf is the mean spectral-centroid difference between adjacent layers (larger is better). "Layer-ID accuracy" is the cross-validated accuracy of a simple frequency-domain classifier that assigns a window to its layer.

PDN decap	Δf (kHz)	Layer-ID accuracy
Low	32	0.92
Med.	26	0.88
High	21	0.84

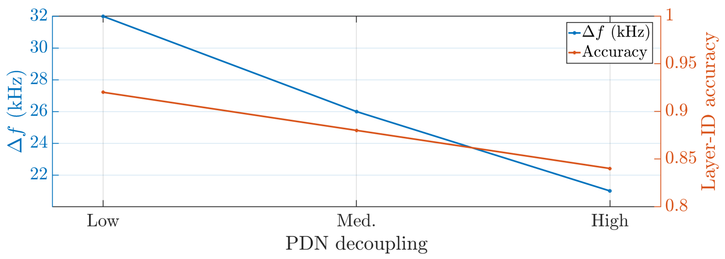


Figure 3.16.: Frequency-domain separability vs. PDN decoupling: Δf (left axis) decreases with stronger decoupling, which also reduces a simple layer-ID classifier's accuracy (right axis).

Shaping with $g_{\mathrm{eval}}(t)/g_{\mathrm{SA}}(t)$ yields

$$\hat{\mathrm{p}}_r(t) = \alpha\, \hat{p}_r^{\mathrm{RC}}\, g_{\mathrm{eval}}(t) + \beta\, \hat{p}_r^{\mathrm{HD}}\, g_{\mathrm{SA}}(t). \tag{3.30}$$

Goodness-of-fit between predicted HD and measured ML energy, and the resulting identifiability, are summarized later in Table 3.10.

Profiling/Collision Workflow

We combine a simple collision strategy with single-trace analysis (SPA): vary a constrained subset of query bits/bytes, record the evaluate-window power or the sense-window peak, and select candidates that minimize power (best match). Iterating this procedure walks toward the stored class hypervector. The algorithm below formalizes the byte-wise variant used in our experiments. End-to-end recovery statistics for different query widths are reported in Table 3.11.

Algorithm 2 Hamming-Distance Collision on CAM-HDC

```
1: procedure CollisionAttack(HDCModel)
2:     for $i \leftarrow 0$ to $n-1$ do                                   ▷ byte index
3:         for $j \leftarrow 0$ to 255 do
4:             queryHV ← 0; queryHV$[0:i] \leftarrow j$
5:             if classify(HDCModel, queryHV) $= c$ then
6:                 candidates$[c][i]$.append($j$)
```

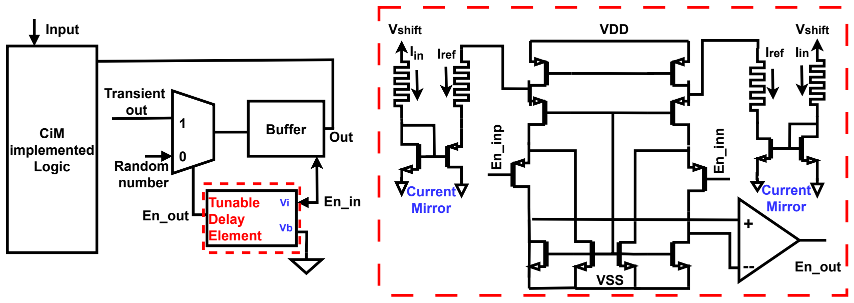


Figure 3.17.: Protected CiM design: tunable delay element in the comparator path to randomize activation timing.

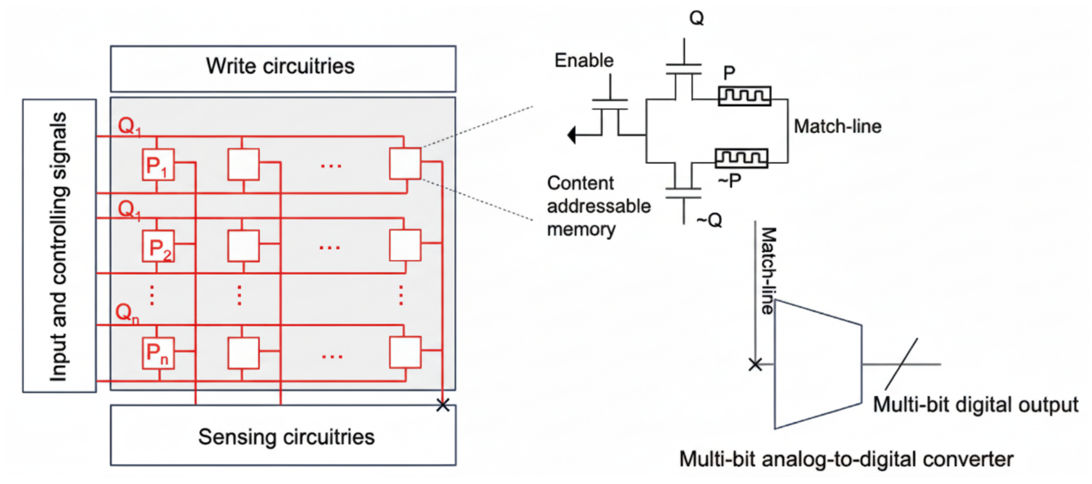


Figure 3.18.: CAM-based HDC search: ML precharge/evaluate and winner-take-all sense.

Table 3.10.: CAM-HDC identifiability: goodness-of-fit between predicted (HD) and measured match-line energy, together with median traces to disclosure for exact class-HV recovery with a 32-byte query. ReRAM exhibits higher amplitude and slightly better model fit, MRAM is cleaner but smaller in magnitude.

Configuration	R^2 (HD $\rightarrow$ Energy)	Median N_{TD}	Notes
ReRAM, Low decap	0.84	280	high amplitude
MRAM, Low decap	0.78	360	cleaner, lower swing
ReRAM, Med. decap	0.80	338	matches Fig. 3.21a trend
MRAM, Med. decap	0.74	410	
ReRAM, High decap	0.76	430	stronger PDN smoothing
MRAM, High decap	0.70	510	

Table 3.11.: Collision-style class-HV recovery for CAM-HDC. Reported are recovered bytes and number of required candidate queries for different query widths. Protected design uses random precharge, staggered activation, and reference dithering.

Setting	**32-byte query**	**16-byte query**	**8-byte query**
Unprotected	28/32 bytes, 338 cand.	16/16 bytes, 156 cand.	8/8 bytes, 68 cand.
Protected (RP+ST+RD)	$\leq$ 1/32 bytes, > 2000 cand.	3/16 bytes, > 2000 cand.	4/8 bytes, > 2000 cand.

Leakage Observation and Results

Figure 3.19 shows that power is minimized for an exact match (smallest HD), and Figure 3.20 confirms the slowest ML discharge in the matched case. Using only 338 candidate traces, the correct class-4 hypervector is identified; the first 32 bytes are recovered with high accuracy (see Figure 3.21). These observations are consistent with the model-fit and disclosure estimates in Table 3.10 and the query-width sweep in Table 3.11.

Countermeasure

Randomize precharge levels and stagger row activation to decorrelate the power-HD mapping across traces. In our protected design, candidates become indistinguishable in evaluate-window power, and the collision attack fails (see Figure 3.22).

Practical Notes. Leakage scales approximately linearly with HD; while the PDN superposes row currents, the winning row produces a distinct correlation signature. Sense-amp jitter broadens the sense window;

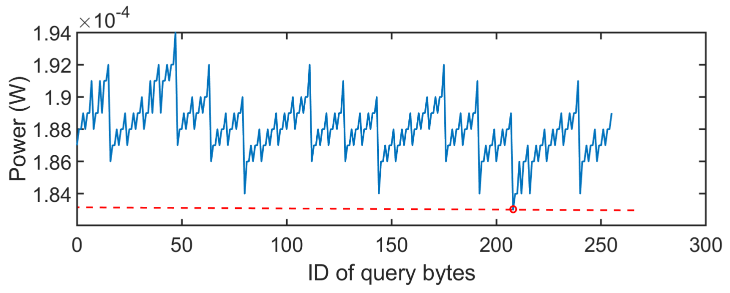


Figure 3.19.: Evaluate-window power vs. 1-byte query candidates: minimum at the correct class hypervector (example: 0xd0).

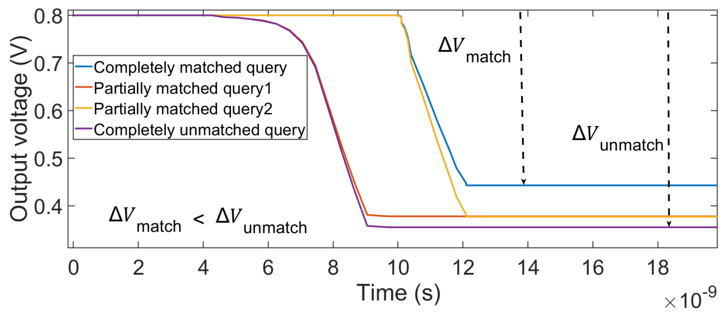


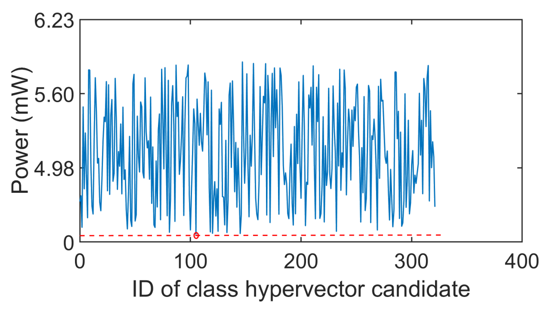
Figure 3.20.: ML voltage: least discharge (closest to V_{pre}) when HD($\mathbf{x}, \mathbf{w}_r$) is minimal (match).

using a wider $g_{\text{SA}}(t)$ improves robustness. Effective mitigations include randomized precharge, activation staggering, and reference dithering.

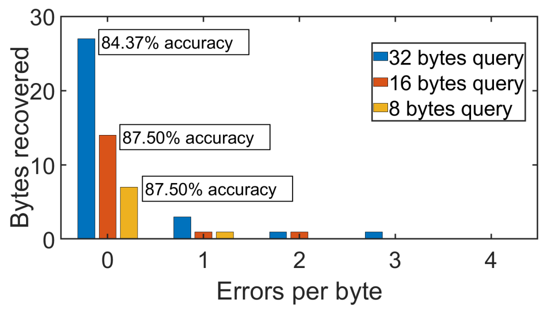
3.3.5. Common Pre-Processing and Evaluation Protocol (used in Cases II-IV)

Pre-processing. We apply baseline removal (polynomial/moving-average detrend), mild band-selection to suppress $1/f$, coarse alignment via cross-correlation with a reference template, optional local time-warp around evaluate/sense edges, and per-window z-score normalization. When noise is colored, a whitened matched filter $w = \Sigma_n^{-1}g/(g^\top\Sigma_n^{-1}g)$ improves SNR in each window.

Decision metrics. Alongside $|\rho|/R^2$, we report traces-to-disclosure N_{TD}, Mahalanobis distances for template separability, per-hypothesis likelihoods $\mathcal{L}$, PCA variance ratios (scree and top-PC scatter), and TVLA (Welch t with $|t|> 4.5$) to corroborate leakage and reproducibility.



(a) Power over top candidates



(b) Bytes recovered vs. query width

Figure 3.21.: Successful collision+SPA attack on class-4 hypervector: correct ID shows the minimum power; byte recovery statistics for 32/16/8-byte queries.

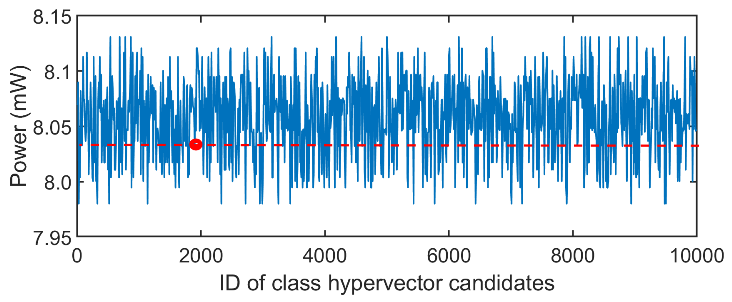


Figure 3.22.: Protected CAM-HDC: power attack fails. Correct and incorrect candidates are indistinguishable.

3.4. Summary

We presented a unified, device-accurate framework for analyzing side-channel leakage in ACIM accelerators. Starting from the fundamental physics of ReRAM and STT-MRAM bitcells, we modeled how conductance variations propagate through the array to create observable power signatures. We dissected four specific compute primitives: Scouting Logic XOR, MAC+tanh, Multi-layer Networks, and CAM-based HDC.

For each primitive, we derived physics-based leakage predictors and demonstrated successful attacks using simulation data. We showed that while standard CPA is effective for simple primitives, advanced techniques like PCA-based layer separation and collision analysis are required for deep networks and high-dimensional search. Finally, we proposed and validated specific circuit-level countermeasures, such as tunable delay elements and dummy-balanced comparators, that effectively suppress leakage with minimal overhead. These results establish a foundation for the secure design of next-generation non-volatile AI accelerators.

4. Vulnerabilities of Emerging NVM-Backed Processor-centric Computing: FPGA Emulation

4.1. Introduction

This chapter bridges the device-level vulnerabilities of STT-MRAM characterized in Chapter 3 (which focused on CiM and sense-path variability) with a *digital* fault-security study. We pivot from analog noise characteristics to a potent digital attack: a **persistent-fault attack on AES when its round-key schedule is stored in MRAM** [109]. We present a practical, board-level, non-invasive voltage-glitch methodology that precisely targets MRAM *write* cycles. By momentarily starving the MRAM's power rail, we bias the spin-transfer-torque switching mechanism to create *persistent* bit-level corruptions in the stored key schedule. We then demonstrate how these stable, reproducible faults enable very low-data-complexity key recovery using DFA and SPFA [9], [69], [82]. We conclude this chapter by proposing several MRAM-specific countermeasures, including both hardware and architectural-level fixes, that can harden designs against this threat class with modest implementation overhead.

Core message. Persistence transforms a one-time physical event into an *algorithmically reusable* cryptanalytic primitive. This is the central thesis of our attack: unlike transient faults (e.g., in SRAM) that must be injected with precise timing for every execution, a single successful persistent-fault injection campaign corrupts the key indefinitely. The adversary can then, at their leisure, collect an arbitrary number of faulty ciphertexts from this single, stable fault. This reuse mechanism is what collapses the data complexity. Under persistence, we demonstrate that AES key recovery becomes possible with only 12–17 correct/faulty ciphertext pairs on real hardware. This result is two orders of magnitude more efficient than many volatile-memory DFA settings, which often require thousands of precisely timed injections and statistical collection to succeed.

Why AES as a case study. Although this experiment targets a cryptographic accelerator, AES serves here as a controlled benchmark to expose a generic persistent-fault primitive in STT-MRAM devices. The motivation is not to study cryptography per se, but to use AES's well-understood algorithmic structure, standardized test vectors, and established differential-fault-analysis framework (DFA/SPFA) as an ideal platform for precisely measuring how device-level, non-volatile write faults propagate through digital logic. This controlled environment allows us to quantify the fault physics and statistical exploitation requirements.

The same non-volatile write mechanism that enables stable key corruption in AES directly threatens AI accelerators that employ MRAM to store weights, feature maps, or hypervectors. The same physical mechanism, a reproducible, persistent bit corruption originating from a glitched MRAM commit window, is equally applicable to MRAM arrays used for storing model weights or hypervectors in AI accelerators. By first demonstrating the fault physics and statistical exploitation on AES, we obtain a reproducible and interpretable baseline. This baseline directly informs the threat and countermeasure models for MRAM-backed AI accelerators, which we explore in the chapter's conclusion. The unified countermeasures proposed, such as PUF-sealing, commit-window randomization, write-verify, and rail monitoring, later reappear as the foundation for securing non-volatile storage in these emerging AI architectures.

4.2. From Analog CiM Variability to Digital Persistent Faults

4.2.1. Why MRAM's Physics Matters for Security

Chapter 3 established that the fundamental MTJ physics enabling STT-MRAM for CiM also yields inherent analog variance, such as read/sense noise and device-to-device asymmetries [43]. Here, we pivot from that analog read-path analysis to a digital write-path threat model. We investigate what happens when MRAM holds sensitive digital keys, and an attacker introduces carefully timed supply glitches during write operations.

By momentarily depressing the supply voltage, we bias the spin-transfer switching process, which can prevent the MTJ's free layer from flipping correctly. This induces non-volatile bit errors that persist across reboots, power cycles, and system resets, remaining stable until explicitly overwritten. This persistence is the key differentiator from transient SRAM/DRAM upsets. These persistent corruptions convert one successful physical fault into a tool for many faulty encryptions. This drastically reduces the ciphertext requirement for DFA and enables the highly efficient SPFA framework.

4.2.2. The Intuition: Persistence $\Rightarrow$ Constancy $\Rightarrow$ Low Samples

The power of persistence is best understood through its direct algorithmic consequence: *constancy*. Let's assume a write-time glitch successfully implants a single-byte difference $\delta \neq 0$ in a round key, for instance in $K_9[j]$, which is now persistently stored. Because this fault is non-volatile, this exact faulty key K_9' is reused for all subsequent encryptions.

Due to the AES structure, this fault in byte j of K_9 propagates through the 10th round's SubBytes and ShiftRows operations to affect exactly one ciphertext byte, $i = \pi^{-1}(j)$, at the input to the final AddRoundKey. For any given plaintext, we can collect the correct ciphertext C (from an uncorrupted key) and the faulty ciphertext C' (from the persistently corrupted key).

For the *correct* guess $k = K_{10}[i]$ of the last-round key byte, the quantity

$$T_k(C[i], C'[i]) \;=\; S^{-1}(C[i] \oplus k) \;\oplus\; S^{-1}(C'[i] \oplus k)$$

will be *constant and equal to* δ across all collected plaintext-ciphertext pairs. This equation effectively "peels off" the last round's AddRoundKey and SubBytes operations, revealing the differential δ at the input to the 10th round's SubBytes. For an *incorrect* key guess $k' \neq K_{10}[i]$, the $T_{k'}$ values will appear random and uncorrelated across the pairs.

This *constancy test* is a powerful distinguisher. It collapses the 2^8 per-byte hypothesis space with very few samples. In an *ideal* model (a perfect single-byte fault, no measurement noise, perfect C/C' pairing), $m = 2$ pairs would suffice to find a unique candidate, and $m = 3$ pairs would make a false positive statistically impossible. On real hardware, however, faults can be messy (multi-byte, multi-bit) and the system has noise. Therefore, robust SPFA scoring under this real-world fault morphology requires about 12–17 pairs to filter out noise and push the per-byte success probability to $\approx$95–100%. As we show later, this high per-byte reliability is necessary to achieve full 16-byte key success.

4.3. Threat Model

Adversary. We assume a non-invasive, board-level attacker who has temporary physical access to a device that stores AES keys in an external or on-chip STT-MRAM. The attacker does not require expensive equipment like a Focused Ion Beam (FIB) or laser. No decapsulation or direct probing of the MRAM die is needed. The attacker also does not modify the AES algorithm or the FPGA's logic. Crucially, the attacker does not need the ability to read the MRAM's contents directly, only to observe the output of the AES engine. The attacker can reset the board, interact with the device using standard I/O (e.g., Universal Asynchronous Receiver-Transmitter (UART), Direct Memory Access (DMA), AXI-Stream), and, most importantly, physically perturb the MRAM's dedicated power rail.

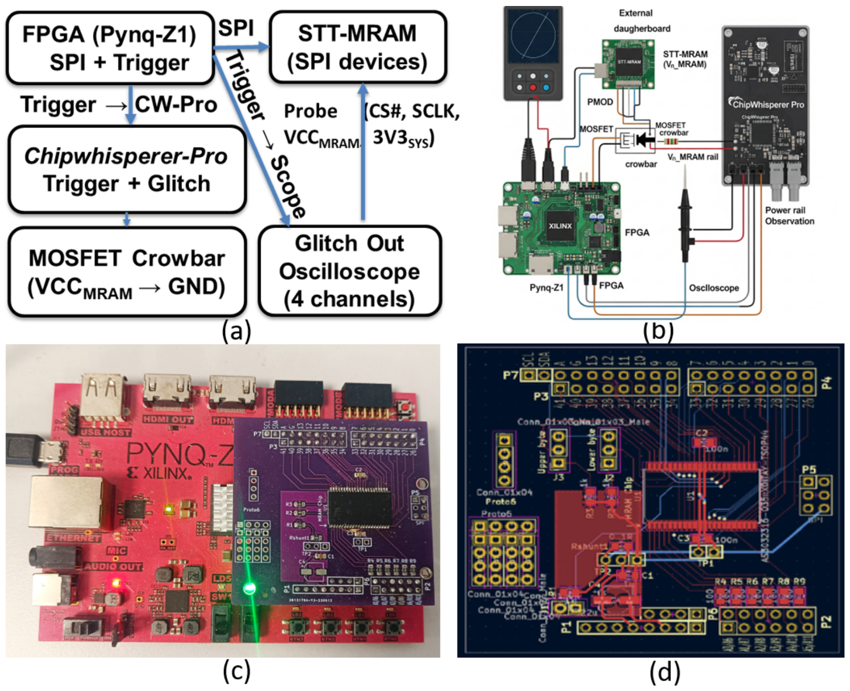


Figure 4.1.: Experimental setup for voltage glitching attacks on STT-MRAM. (a) System-level architecture showing FPGA (Pynq-Z1), ChipWhisperer-Pro, and oscilloscope connections for glitch injection and monitoring. (b) Laboratory setup with MRAM mounted on Pynq-Z1, ChipWhisperer CW305, and CW1200. (c) Hardware implementation with MRAM daughterboard interfaced to the Pynq-Z1 FPGA. (d) PCB layout of the custom MRAM daughterboard.

Capabilities. The attacker possesses three key capabilities: (i) **Voltage glitching:** The ability to connect to the V_{MRAM} power rail (e.g., at a decoupling capacitor) and momentarily short it to ground using an external MOSFET crowbar, inducing a voltage droop. (ii) **Chosen I/O:** The ability to send plaintexts to the AES engine and receive the corresponding ciphertexts. This is a standard user-level interaction. (iii) **Coarse timing:** The ability to roughly synchronize the glitch with the MRAM write operation. This does not require picosecond precision; monitoring a coarse signal like a GPIO pin, an LED, or a serial "Key-loading..." message is sufficient to align the glitch within the millisecond-scale window of the key-writing process.

Objective. The attacker's goal is to induce a *persistent* byte difference δ in the MRAM-resident expanded AES key. Once this stable fault is implanted, the attacker will perform a "pre-fault" encryption (to get C) and a "post-fault" encryption (to get C') using the same plaintext. By collecting a small number of these (C, C') pairs, the attacker aims to recover the full 128-bit master key via the DFA/SPFA constancy test.

4.4. Experimental Platform and Instrumentation

4.4.1. Board-Level Setup

We use a commodity, off-the-shelf Xilinx Pynq-Z1 FPGA board, which features a Zynq-7000 System-on-Chip (SoC). This board is connected via its Peripheral Module Interface (PMOD) header to a custom SPI STT-MRAM daughterboard (featuring a commercially available MRAM chip). To enable clean glitch injection, the MRAM's power rail (V_{MRAM}) is isolated from the board's global 3.3 V rail using a small series element (either a 1–2 Ω resistor or a ferrite bead). This isolation prevents the glitch from crashing the main Zynq processor while ensuring the MRAM chip feels the full effect of the droop.

A ChipWhisperer-Pro serves as our glitch controller, driving a low-$R_{\mathrm{DS(on)}}$ MOSFET configured as a "crowbar." When triggered, the MOSFET briefly shorts V_{MRAM} to ground, generating sharp 10–80 ns voltage droops of 350–700 mV. For validation, a Tektronix MSO 2024B oscilloscope monitors V_{MRAM}, the main 3V3_SYS rail, the SPI chip select (CS#), SPI clock (SCLK), and the glitch trigger (TRIG) signal. Figure 4.1 shows the overall system diagram and lab photos that we will reference during timing analysis.

4.4.2. AES Integration and Key-Schedule Handling

To create a realistic but faultable interface, we designed our system to explicitly store all AES round keys (K_0 through K_{10}) in the external STT-MRAM, rather than caching them in the FPGA's on-chip Block RAM (BRAM) as a fully hardened design might. We implemented a custom Key-Schedule Manager (KSM) module in the FPGA logic. This KSM is responsible for streaming the key bytes from the MRAM over the Serial Peripheral Interface (SPI) bus into the AES core as needed. The AES core itself remains unmodified (a standard, unhardened AXI-Stream-based core). This architecture exposes the key-schedule-to-MRAM interface, making the write operation a distinct, targetable event.

4.5. Attack Methodology: Targeting the MRAM Commit Window

4.5.1. Timing Model and Parameter Sweep

Our first step is to precisely locate the MRAM's sensitive commit window. We know that MRAM write cycles are significantly longer and more power-intensive than read cycles. Only the narrow *commit window* within this cycle is sensitive to voltage faults. To find it, we generate one trigger (TRIG) signal from the FPGA at the start of each SPI data write. We then systematically sweep the glitch parameters:

- **Offset:** from -200ns before the TRIG to +800ns after.
- **Width:** from 5ns to 200ns.
- **Depth:** from 10% to 35% drop of V_{MRAM}.

For each (offset, width, depth) tuple, we perform ≥1000 write-and-read-back repetitions to build a statistical map of the fault probability. As shown in Figure 4.2, the device-interface timing clarifies why the write path is vulnerable, and the empirical sweep in Figure 4.3a pinpoints the narrow high-probability region (validated by the scope captures in Figure 4.3b).

4.5.2. Persistence Definition and Morphology

Not all observed faults are useful. We establish a strict definition for a *persistent fault*: a corrupted byte must (i) appear identically across 100 consecutive read-back attempts, (ii) survive ≥5 complete board power cycles (cold reboots), and (iii) be correctly cleared upon a subsequent, clean (non-glitched) overwrite.

When analyzing the faults that meet this definition, we observe their fault morphology (i.e., their physical shape). Spatially, the faults are highly localized: we observe 1–3 flipped bits, almost always within a single byte, with only rare spillover to an adjacent byte. Temporally, we confirm the STT asymmetry: the P→AP transition ('0'→'1') shows up to 3× higher susceptibility to glitches than the AP→P transition. This physical bias, shown in Figure 4.4, is a key finding we can use to optimize the SPFA.

4.6. Cryptanalysis Under Persistent Faults

4.6.1. Notation and Fault Model

We formalize the attack on AES-128 [7]. Let the round keys be $K_0, \ldots, K_{10}$. We use S for the SubBytes S-box and S^{-1} for its inverse. π denotes the ShiftRows permutation, and $x[i]$ denotes byte i of a state x. Our persistent fault model states that a successful glitch injects a fixed, nonzero byte difference $\delta \in \mathbb{F}_{2^8}$

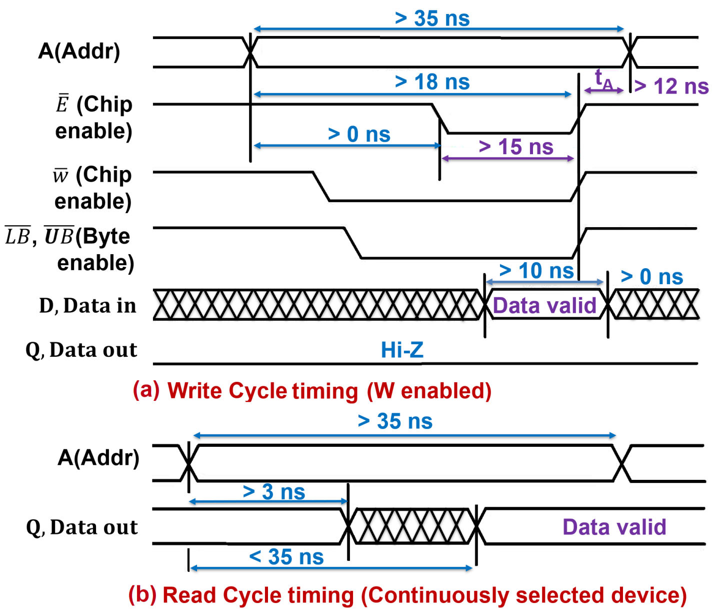
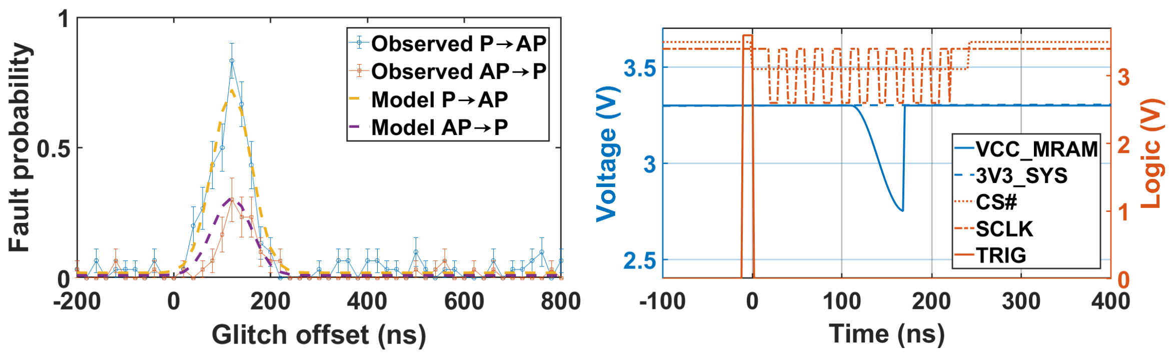


Figure 4.2.: Timing diagrams at the device interface, showing longer write cycle requirements compared to read access. The extended write window makes write operations more susceptible to precisely timed glitch injection, which is leveraged in our fault attack model.



(a) Glitch window characterization: measured fault probability vs. glitch offset for P→AP and AP→P transitions.

(b) CS#, SCLK, and TRIG traces confirm that the glitch overlaps the MRAM write commit window.

Figure 4.3.: Empirical characterization and validation of optimal glitch injection timing in MRAM writes.

into some round key byte $K_r[j]$. This creates a faulty key schedule K' where $K'_r[j] = K_r[j] \oplus \delta$, and this K' is used for *all* subsequent encryptions until it is overwritten.

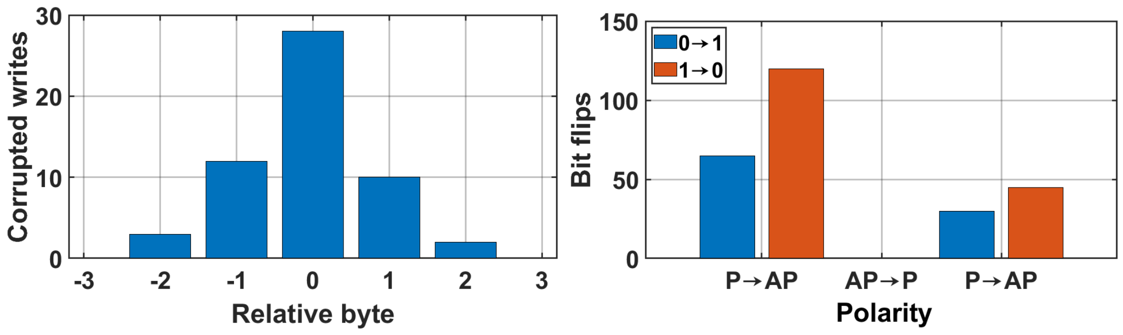
4.6.2. Case A: Persistent Fault in K_{10} (Direct Leakage)

This is the simplest, though less common, case. If the attacker successfully glitches a byte i in the *final* round key, $K'_{10}[i] = K_{10}[i] \oplus \delta$, then for any plaintext P, the correct and faulty ciphertexts C, C' will be:

$$C'[i] = S(X_9[i]) \oplus K'_{10}[i] = S(X_9[i]) \oplus K_{10}[i] \oplus \delta = C[i] \oplus \delta$$

$$C'[j] = C[j] \quad (\text{for } j \neq i)$$

The fault difference $\Delta C[i] = C[i] \oplus C'[i] = \delta$ is observed directly on the ciphertext output. This immediately exposes the corrupted byte location i and the fault value δ. Standard last-round differential tests can then be used to recover $K_{10}[i]$ immediately.



(a) Persistent faults: spatial locality (targeted byte vs. neighbors). **(b)** Persistent faults: bit-flip polarity with P/AP asymmetry.

Figure 4.4.: Spatial/polarity characterization of persistent MRAM faults, used later to weight SPFA scores.

4.6.3. Case B: Persistent Fault in K_9 (Constancy Test)

This is the more general and powerful attack. Let the persistent fault be $K_9'[j] = K_9[j] \oplus \delta$. This fault δ at the input to round 9 propagates. At the input to the last round (after round 9's SubBytes, ShiftRows, and MixColumns), the difference will have diffused. However, when we look at the *output* ciphertext, only byte $i = \pi^{-1}(j)$ (the byte affected by ShiftRows from j) is impacted by this fault *before* the final MixColumns (which doesn't exist in the last round). Let $u[j]$ be the state input to 10th-round SubBytes.

$$C[i] = S(u[j]) \oplus K_{10}[i]$$

$$C'[i] = S(u[j] \oplus \delta) \oplus K_{10}[i]$$

To find the unknown $K_{10}[i]$, we test all possible values k in our case. We rearrange the equations: $u[j] = S^{-1}(C[i] \oplus k)$ and $u[j] \oplus \delta = S^{-1}(C'[i] \oplus k)$. Eliminating the unknown state $u[j]$ by XORing the two equations gives the PFA constancy test:

$$T_k(C[i], C'[i]) \;=\; S^{-1}(C[i] \oplus k) \;\oplus\; S^{-1}(C'[i] \oplus k) \;=\; \delta \quad\Longleftrightarrow\quad k = K_{10}[i]. \tag{4.1}$$

For the correct key k, T_k will equal the *constant* value δ for all pairs (C, C'). For a wrong key k, T_k will be random. The chance that a wrong key k *also* produces a constant T_k across m pairs is approximately $2^{-8(m-1)}$. Using $m = 3$ pairs makes the error probability $\approx 2^{-16}$, making false positives negligible. *This is an analytic lower bound* [9].

4.6.4. From Per-Byte to Full-Key: Required Reliability

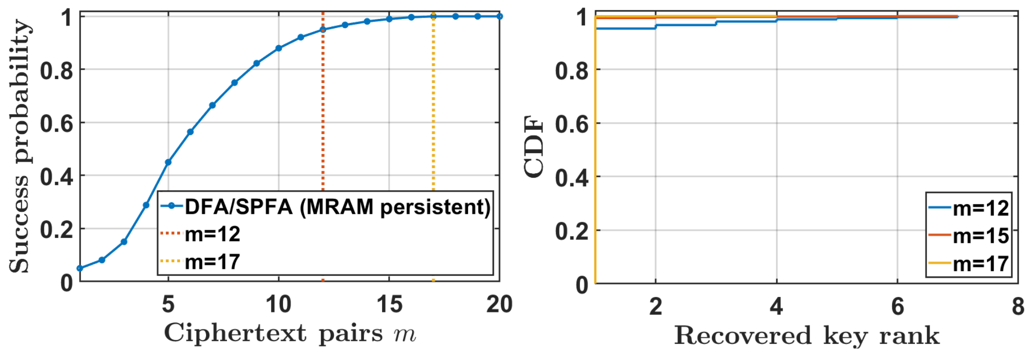
The constancy test recovers one byte, $K_{10}[i]$, at a time. To recover the full 16-byte key K_{10}, we need all 16-byte tests to succeed. Define $p_{\text{byte}}(m)$ as the success probability for a single byte using m pairs. Assuming the byte recoveries are approximately independent, the probability of recovering the *entire* key is:

$$p_{\text{all}}(m) \approx (p_{\text{byte}}(m))^{16}$$

This formula reveals a critical challenge. Even if our per-byte success is a high 95% ($p_{\text{byte}} = 0.95$), the full-key success would be $0.95^{16} \approx 0.44$ (less than 50/50). To target a robust full-key success of $p_{\text{all}} \geq 0.95$, we require a per-byte success of $p_{\text{byte}} \geq 0.95^{1/16} \approx 0.9968$. This explains why our experiments must push m well beyond the ideal 2–3 pairs, into the 12–17 range, to achieve near-certain full-key success on real, noisy hardware.

4.6.5. Robustness: SPFA Scoring

Real-world faults are not always ideal single-byte injections. Multi-byte bursts (~6% of our campaigns) and the P→AP asymmetry distort the pure constancy test. For example, a multi-byte fault might mean



(a) Per-byte success probability vs. number of pairs **(b)** Key-rank CDF for one byte of K_{10}

Figure 4.5.: DFA/SPFA effectiveness under persistent faults: (a) ≈12 pairs reach ~95% per-byte success; (b) with m=17, the correct key reaches rank-1 in > 99% of trials.

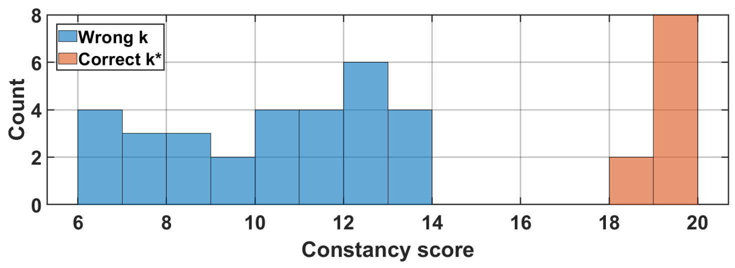


Figure 4.6.: SPFA score distribution: the correct key k^* yields a sharp, stable peak at the same δ across pairs; wrong keys are near-uniform.

that T_k is constant for 90% of pairs but "noisy" for 10%. A simple "is-it-constant" check would fail. We therefore use a more robust SPFA-style consistency score. Instead of a binary check, we compute the histogram of $T_k^{(t)}$ values across all m pairs for a given key k. The correct key k will have a histogram with a very strong peak at δ, while wrong keys k' will have flat, uniform histograms. This scoring method tolerates partial/noisy constancy and still singles out the correct key. This score can be optionally weighted by physically likely δ values (e.g., those corresponding to P⟶AP flips). The characteristic separation this produces is shown later in Figure 4.6.

4.7. Results and Analysis

4.7.1. Selective Glitching Without Board Destabilization

Our oscilloscope captures confirm the effectiveness of the isolated power rail design. We successfully generated sharp 350-700mV droops on V_{MRAM}, lasting 10-80ns. Critically, the main 3V3_SYS rail for the Zynq processor remained stable, deviating by < 20mV. This proves that we can aggressively fault the MRAM without causing a system-wide crash or reset, a key requirement for a practical attack. Faults were only observed when the droop pulse overlapped with the pre-characterized commit window (from Figure 4.3a and Figure 4.3b), validating both our rail isolation and timing alignment methodology.

4.7.2. Glitch Window and Asymmetry

The experimental results from the parameter sweep (section 4.5) are clear. The high-probability fault window is consistently ≈40–60ns wide and centered ≈+120ns after the SPI write trigger (Figure 4.3a). Scope validation in Figure 4.3b shows the overlap of the crowbar pulse with this commit window. We also quantified the physical asymmetry. By controlling the data written (all '0's vs. all '1's), we isolated P⟶AP and AP⟶P transitions; the former reached 55–70% peak fault probability vs. 20–35% for the latter, which is consistent with the polarity histograms in Figure 4.4b.

Table 4.1.: Our work vs prior MRAM security literature

Works	**Platform**	**Attack Class**	**Result (AES)**
Reliability/Endurance of STT-MRAM [61], [68]	Simulation	Variation / stress analysis	N/A
Leakage in MRAM systems [58]	Device	Side-channel (timing/power)	N/A
Magnetic field bias on pMTJ MRAM [71], [76]	Device	Magnetic fault/bias	N/A
CPA on MRAM-backed crypto [50]	Device + board	Power side-channel (CPA)	key recovery 2k–5k traces
This work (MRAM write-cycle faults)	Device + board	Voltage-glitch during writes	key recovery 12–17 ciphers

4.7.3. Persistence and Morphology

We validated the non-volatility of the faults. All injected faults that passed our filter remained perfectly stable across 100 consecutive reads and, more importantly, across at least five full power cycles of the board. They were only cleared by a clean, non-glitched write. Out of 50 successful fault-injection campaigns, 42 produced a persistent fault. The morphology was predominantly byte-local: most persistent faults consisted of 1–3 flipped bits within a single target byte. However, multi-byte bursts (corrupting 2–4 adjacent bytes) occurred in ~6% of campaigns, reinforcing the need for robust SPFA scoring over simple DFA as visualized by Figure 4.6.

4.7.4. Key-Recovery Effectiveness

With persistent K_9 faults, we evaluated the SPFA algorithm's effectiveness.

- **Per-byte success vs. pairs:** Using our SPFA scoring, acquiring $\approx$*12 pairs* yields a ~95% per-byte success rate (Figure 4.5a). Pushing to $\approx$*17 pairs* increases the per-byte success to ~100%, and the correct key reaches rank-1 in >99% of trials (Figure 4.5b).
- **SPFA separation:** A visualization of the SPFA scores (Figure 4.6) shows the correct key's score clustering tightly at the maximum, while all 255 wrong key guesses have scores scattered near-uniformly, demonstrating a large statistical separation.

Because each (C, C') pair contains information for *all* 16 bytes (as the fault propagates differently to each), the same $m = 12$–17 pairs are sufficient for the full-key recovery.

4.7.5. Why 12–17 on Hardware (vs. 2–3 in the Ideal Model)

The discrepancy between the analytic bound ($m = 2$ or 3) and our hardware result ($m = 12$ to 17) is explained by real-world non-idealities: (i) **Fault Morphology:** The ideal model assumes a single-byte fault δ. Our hardware sometimes produced multi-byte faults or δ values that were not perfectly constant (e.g., if the glitch was probabilistic), creating "noise" in the constancy test. (ii) **Statistical Robustness:** The SPFA scoring method, which uses a histogram, inherently requires more samples than a simple binary check to build a statistically significant peak. (iii) **Full-Key Requirement:** As shown in section 4.6, achieving $p_{\text{all}} \geq 0.95$ requires a per-byte reliability $p_{\text{byte}} \gtrsim 99.7\%$. The ideal $m = 3$ might only give $p_{\text{byte}} = 99\%$. The extra pairs (from 3 to 17) are needed to bridge this gap and ensure all 16 bytes are recovered reliably.

4.7.6. Comparison to Volatile-Memory DFA

The impact of this result is best seen in comparison. Classical DFA on volatile memory (SRAM/DRAM) is a high-effort attack. The fault is transient, so the attacker must re-inject a fault *for every single ciphertext* they wish to collect. This often requires thousands of attempts, meticulous synchronization, and complex statistics to separate successful injections from system crashes. Our persistent-fault method requires *one* successful glitch campaign. After that, the per-run synchronization challenge is eliminated. This collapses the data complexity and practical effort to just 12–17 pairs for reliable, full-key recovery.

4.8. Ablations and Sensitivity

4.8.1. Targeting K_{10} vs. K_9

We analyzed attacks targeting both K_{10} and K_9. Targeting K_{10} is simpler, as it yields direct byte differences $\Delta C = \delta$ (Case A). This is easy to detect and requires fewer algebraic checks. However, this simplicity is also a weakness, as the direct ΔC can be easily masked by simple countermeasures. Targeting K_9 (Case B) is more powerful. It unlocks the stronger, non-linear constancy test of Equation 4.1. We found this test performs much better under the robust SPFA scoring, especially when the fault morphology is messy (e.g., multi-byte), as the S-box's non-linearity helps differentiate the signal.

4.8.2. Effect of Multi-Byte Bursts

We explicitly studied the ~6% of campaigns that resulted in multi-byte fault bursts. When bursts flip adjacent round-key bytes (e.g., $K_9[j]$ and $K_9[j+1]$), the diffusion of AES causes some ciphertext bytes to carry overlapping evidence from both faults. A simple DFA test would fail. However, SPFA's aggregate histogram-based score is highly resilient. It effectively integrates the "partial evidence" from all pairs and bytes, reducing false positives and still identifying the correct key, albeit at the cost of requiring a slightly higher m (e.g., closer to 17 pairs than 12).

4.8.3. Plaintext Selection

We tested whether chosen-plaintext strategies (e.g., selecting plaintexts that stress specific S-box activity patterns or known differential trails) offered any advantage over uniform random plaintexts. We found that uniform random plaintexts are perfectly sufficient. The "signal" from the persistent fault δ is so strong and consistent that it dominates the statistical analysis. While chosen-plaintext strategies might offer a marginal gain (e.g., reducing m from 17 to 16), the empirical improvement was within the noise bound of our experimental campaigns.

4.9. Countermeasures Tailored to MRAM

Our analysis of the fault's physical origin (the commit window) and its algorithmic exploitation (persistence) points to several effective, MRAM-specific countermeasures.

4.9.1. PUF-Bound Sealing of Key Slots

We bind each MRAM key slot to a device-unique secret derived from a Physical Unclonable Function (Physical Unclonable Function (PUF)). On boot, the device reads the MRAM slot, decrypts it with the PUF key, and verifies an integrity tag (e.g., a MAC). A persistent corruption from a glitch attack will cause this integrity check to fail. The system can then securely invalidate the faulty slot and request re-provisioning. This countermeasure converts the *silent* data corruption into a detectable attestation failure, preventing any reuse of the faulty key.

4.9.2. Randomized Commit Timing (Write Dither)

Our attack relies on a fixed, predictable glitch offset (+120ns) to hit a fixed, predictable commit window (40-60ns wide). This can be broken by introducing jitter. A hardware write-controller can dither the MRAM write pulse timing. By adding a PUF-keyed, pseudo-random delay of $\pm(80\text{–}100)$ ns to the start of the commit, the sensitive window will move around randomly in time, far exceeding its 40-60ns width. An attacker's fixed-offset glitch will now miss the window most of the time, reducing the fault probability from >50% to near zero. This adds negligible latency overhead.

4.9.3. Dual-Slot + ECC/CRC + Write-Verify

This is a classic data-integrity solution, applied here for security. Instead of storing one copy of the key, we store two masked copies per slot (e.g., K and $K \oplus R$) and/or add Error-Correcting Code (ECC) or a Cyclic Redundancy Check (CRC). Most importantly, implement a write-verify protocol: after every MRAM write, the KSM must immediately read the data back and verify its integrity (checking ECC/CRC, or that the two masked copies XOR to R). This check must happen before the key is used. Our dominant fault morphology (single-byte flips) is easily caught by this. On mismatch or ECC failure, the system must scrub and rewrite the slot.

4.9.4. Rail Monitoring and Write Gating

Since the attack requires a fast voltage droop on V_{MRAM}, this droop can be detected. An on-chip, lightweight monitor (e.g., a fast ring-oscillator (RO) or the Zynq's built-in XADC) can sense the V_{MRAM} rail. The MRAM write-controller can be gated by this monitor: it should refuse to initiate a write cycle unless the rail has been detected as stable for a minimum pre-window duration. This blocks a large fraction of simple crowbar-style attempts or forces the adversary into using much smaller, less-effective (and thus less reliable) glitch pulses.

4.9.5. Architectural Surface Reduction

The most robust solution is to reduce the attack surface entirely. Avoid storing the full, expanded round-key schedule persistently. Instead, store only the 128-bit master key (sealed and PUF-bound, as above). The round keys can then be generated on-the-fly by a hardware key-expansion core and stored in volatile, on-chip BRAM for each encryption session. This eliminates the MRAM write operation on the sensitive round keys entirely.

4.10. Discussion and Outlook

4.10.1. Why This Work is Effective

This attack is highly effective because it represents a confluence of three factors: (1) **Physically grounded:** It is not a theoretical model. It exploits a narrow, repeatable, physical property of STT-MRAM: the write-cycle commit window and its STT asymmetry. (2) **Algorithmically potent:** It leverages the *persistence* of the NVM to perfectly match the PFA/SPFA attack model, which provides the constant-δ

structure needed for the constancy test. (3) **Practical:** The attack is board-level, non-invasive, and uses commodity tools (FPGA, ChipWhisperer). It achieves a full key recovery with a remarkably low data complexity of just 12–17 pairs.

4.10.2. Beyond This Chapter: Invertible Randomized Wrappers

Looking forward, a more advanced architectural hardening could involve per-encryption invertible nonlinear transforms. The idea is to insert byte-wise bijections (e.g., small, keyed S-boxes) as pre- and post-processing wrappers around the standard AES core. These wrappers would be re-keyed for every encryption. This breaks the fundamental assumption of PFA/SPFA: even if the fault δ in the key is persistent, the observable differential at the ciphertext $T_k(C, C')$ is no longer constant, as it is "de-correlated" by the changing wrappers. Note that simpler countermeasures like linear masks or masked S-boxes (common in SCA defense) are often ineffective against this attack, as the persistence-driven constancy holds regardless of linear masking.

4.11. Practical Deployment Guidance

Based on these findings, we offer concrete guidance for engineers deploying MRAM-backed secure systems:

- **Supply routing:** Treat V_{MRAM} as a sensitive security rail. Isolate it from other rails (as we did) and add local monitoring. Avoid routing externally visible trigger signals (like ”SPI_CS”) that correlate directly with MRAM commit windows.
- **Key lifecycle:** All persistent key slots must be sealed (e.g., to a PUF) and tagged with version numbers or integrity MACs. Implement a secure boot process that scans and verifies these slots. Periodically scrub and refresh slots.
- **Helper data:** If using PUFs or ECC, the helper data and ECC tags must also be protected with integrity checks. Never store raw, reconstructible PUF responses.
- **Secure boot:** The bootloader must "fail closed." On any MRAM slot verification failure, it must halt, log the event, and enter a secure recovery or re-provisioning mode, rather than booting from a potentially corrupted state.

4.11.1. Implications for Emerging AI Accelerators

Although the quantitative analysis in this chapter targets an AES key schedule, the primary motivation is to characterize a fundamental vulnerability. The underlying threat is not specific to AES; it is the vulnerability class of write-time, byte-local persistent corruption in STT-MRAM. This exact primitive has direct and severe implications for emerging AI accelerators that increasingly rely on MRAM and other NVMs for on-device model storage, in-memory computing, or intermediate caching. Three concrete mappings illustrate this risk transfer.

First, MRAM is increasingly proposed for direct weight and hypervector storage due to its non-volatility, high density, and low-leakage (zero standby power). A persistent bit flip in an on-device neural network weight or a Hyperdimensional Computing (HDC) class hypervector creates a *persistent* change in the model's inference behavior. This is perfectly analogous to a persistent key fault producing repeatable faulty ciphertexts. In HDC systems, where classification relies on bit-wise similarity (e.g., Hamming distance), a few targeted, flipped bits in a stored class hypervector could systematically alter similarity scores, inducing a targeted and repeatable misclassification (e.g., forcing a specific input to always be classified as an adversary-chosen category).

Second, modern secure deployment practices often mandate that on-device models are encrypted at rest. The decryption key for these models must be stored somewhere. If this key is stored in MRAM, it becomes a direct target. Our SPFA demonstration on MRAM-backed AES shows that such key material is highly vulnerable to low-cost, low-sample persistent fault extraction. Once this model-decryption key is recovered, the proprietary AI model (the weights, architecture) becomes trivially exfiltrable, constituting a major intellectual property breach. Thus, SPFA is not merely a cipher-level concern but a direct threat to model confidentiality.

Third, hybrid attacks become possible. An adversary could use passive side-channel analysis (SCA) to profile the accelerator's memory access patterns, identifying the precise timing and physical addresses of sensitive write operations (e.g., model update or key provisioning). Armed with this timing information, they can then launch a targeted persistent-glitching attack, maximizing the probability of corrupting a specific, high-value parameter. This combined SCA+PFA pipeline is particularly potent for edge AI accelerators that operate with real-time constraints and often forgo robust write-verification checks to save power and latency.

These observations motivate a unified defensive strategy. The same security principles apply. Measures that defend MRAM-backed AES, such as PUF-sealing of storage slots, randomized commit timing, write-verify with ECC/CRC, and on-chip rail monitoring, naturally transfer to securing AI accelerators. We advocate for encrypting and/or MAC-ing model parameters under a PUF-derived key, using hardware to jitter MRAM write timing to prevent repeatable fault injection, implementing robust post-write detection (write-verify) to detect and scrub persistent corruptions, and avoiding long-lived persistent caches of any sensitive model artifacts. We demonstrate the foundation for this transferability in section 4.9 by applying PUF-binding and write-verify to the AES setting, outlining how these identical measures protect model integrity and confidentiality in broader AI accelerator contexts.

4.12. Summary

This chapter demonstrated that MRAM's core feature—non-volatility—can be turned against it. A well-timed, non-invasive write-cycle glitch becomes a *persistent* cryptanalytic primitive. This primitive creates stable, byte-localized key-schedule faults that survive reboots. We showed how this persistence perfectly matches the PFA/SPFA model, collapsing the data complexity for a full AES-128 key recovery to a practical 12–17 ciphertext pairs on real hardware. We presented the physically grounded methodology for finding the MRAM commit window, a complete cryptanalysis (using the Equation 4.1 constancy test and robust SPFA scoring, Figure 4.6), and a suite of MRAM-specific countermeasures (like write-dithering and PUF-sealing) with modest overheads. The results send a clear message: MRAM-backed cryptosystems and secure storage designs demand a new class of defenses that go beyond those designed for transient SRAM/DRAM faults, with direct implications for the security of next-generation AI accelerators.

5. Vulnerability of Flexible Edge-AI-based Neuromorphic Computing: Simulation

5.1. Introduction

The convergence of AI with novel hardware form factors is paving the way for ubiquitous, intelligent systems. Applications in wearable health monitoring, soft robotics, and bio-electronic interfaces demand computing platforms that are not only energy-efficient but also mechanically compliant, lightweight, and low-cost. FE, often based on TFTs like amorphous indium-gallium-zinc oxide (a-IGZO), have emerged as a primary enabler for this vision, allowing computation to be integrated directly onto conformal and even transparent substrates [101].

In parallel, biologically inspired SNNs have garnered significant interest as a power-efficient computing paradigm for these edge applications. Unlike conventional Artificial Neural Network (ANN)s, SNNs operate on event-driven, discrete spikes, performing sparse, asynchronous computation. This model drastically reduces redundant switching activity, making f-SNNs an ideal match for the stringent power and thermal budgets of on-skin or implantable devices.

However, this combination of flexible substrates and event-driven computing introduces a new, largely unexplored security vulnerability. The very attributes that make FE attractive are, from a security perspective, critical weaknesses:

- **Reduced Physical Protection:** Flexible devices inherently lack the rigid packaging, metallic shielding, and complex multi-layer ground planes of conventional silicon chips. This thin encapsulation significantly increases exposure to physical leakage.
- **Simplified Power Delivery:** To maintain low cost and flexibility, these systems often rely on limited I/O and shared power supply rails, restricting the use of common side-channel defenses like split-domain power or on-chip filtering.
- **Data-Correlated Activity:** The event-driven nature of SNNs, while efficient, directly correlates power consumption with the data being processed. Each spike, or lack thereof, represents a discrete computational event, yielding current and timing signatures that are rich with information.

This research confronts the hypothesis that these factors combine to make flexible SNNs (f-SNNs) highly susceptible to power-based SCA. While SCA on rigid CMOS neural accelerators is a well-established field [86], the unique device physics of a-IGZO TFTs and the analog, continuous-time dynamics of flexible circuits present a distinct and uncharacterized threat landscape [95].

To bridge this critical gap, this chapter introduces *FlexSpy*, a comprehensive, design-time side-channel framework specifically engineered for flexible neuromorphic hardware. *FlexSpy* provides the first end-to-end pipeline to model, simulate, and quantify power-based information leakage in f-SNNs *before* fabrication.

5.2. Hardware Design

5.2.0.1. Flexible Spiking Neural Networks (f-SNN).

Spiking neurons [97], [98], [102] exchange discrete events and remain quiescent between spikes. As shown in Figure 5.1 (left), a typical f-SNN cell integrates (i) a synaptic input stage that forms a current or conductance sum

$$i_{\text{syn}}(t) = \sum_i w_i\, s_i(t),$$

(ii) an *RC* integrator that accumulates charge to a threshold, and (iii) reset-discharge control that enforces a refractory interval. These blocks map well to TFT physics and enable low-power operation on bendable substrates.

However, their continuous conduction paths and shared supply rails create data-dependent shifts in the supply current: quasi-DC offsets during spike epochs with sharp edges at spike onsets/offsets. These offsets later provide alignment anchors and classification features in side-channel traces, as exploited by FlexSpy.

5.2.0.2. Flexible Recurrent Neural Networks (f-RNN).

Flexible recurrent designs realize smooth state dynamics without discrete spikes. Each hidden state is stored on a capacitor, while first- or second-order *RC* sections set learnable time constants $\tau_i = R_i C_i$. Weighted sums and activation functions use resistive networks and transconductance stages, forming a continuous-time approximation to recurrent models, as shown in Figure 5.1 (right). Compared with sparse, event-driven f-SNNs, f-RNNs produce smoother supply-current envelopes with weaker spike-aligned features; leakage concentrates at low frequencies and reflects interactions between state voltages, bias currents, and the power-delivery network (PDN).

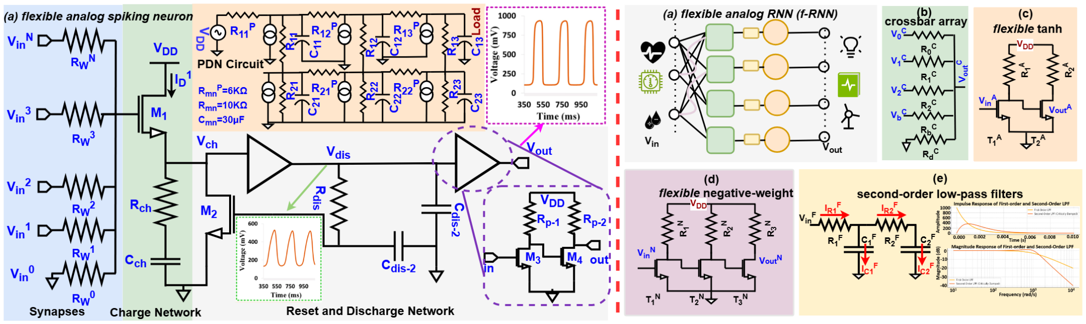
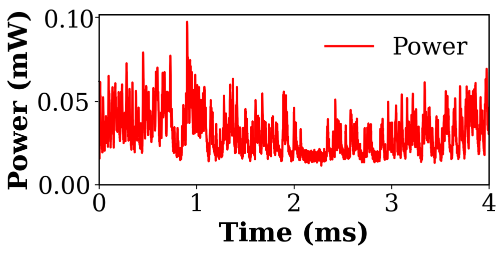

(a) flexible analog spiking neuron
PDN Circuit
Synapses
Charge Network
Reset and Discharge Network
(a) flexible analog RNN (f-RNN)
(b) crossbar array
(c) flexible tanh
(d) flexible negative-weight
(e) second-order low-pass filters
Time (ms)
Voltage (mV)

Figure 5.1.: Circuit-level implementations of flexible neuromorphic circuits (f-NCs) analyzed in this work. Left: *f*-SNN cell with Synapse, Charge/Integrate, and Reset/Discharge stages. Right: *f*-RNN cell with recurrent *RC* dynamics for continuous-time state evolution.

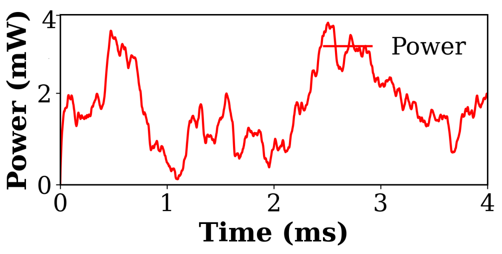
As summarized in Figure 5.2, the f-SNN supply current exhibits step-like, spike-synchronous offsets, whereas the f-RNN output is smoother and less sharply aligned to individual events. This qualitative difference is the root cause of richer, more easily exploitable leakage in f-SNNs.

5.3. The FlexSpy Framework Methodology

FlexSpy is a design-time, simulation-based framework for predicting, localizing, and mitigating power side-channel leakage in flexible neuromorphic circuits (f-NCs). The complete end-to-end procedure is illustrated conceptually in Figure 5.3 and detailed in Algorithm 3. The pipeline begins with device models and a circuit netlist, synthesizes power traces, extracts spike-aligned features, and then executes a calibrated attack suite to quantify and localize leakage.



(a) Raw traces of f-SNN showing spike-triggered current steps.



(b) Raw traces of f-RNN showing smooth oscillatory envelopes.

Figure 5.2.: Distinct leakage primitives from raw power traces: spike-driven quasi-DC offsets in f-SNN (left) vs. smoother low-frequency RC oscillations in f-RNN (right).

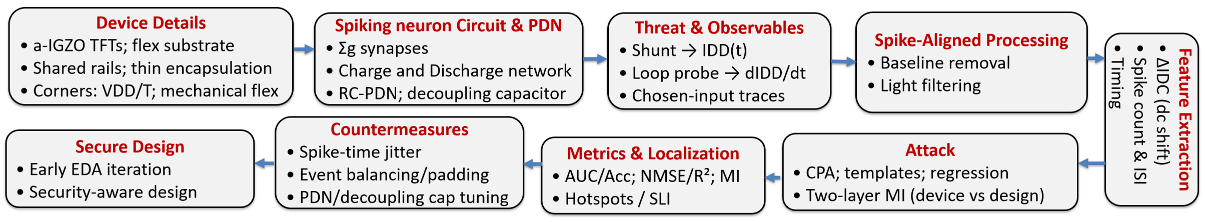


Figure 5.3.: Overview of the FlexSpy framework pipeline. FlexSpy provides a complete design-time flow: from technology-calibrated device simulation and PDN modeling to spike-aligned feature extraction, a calibrated attack suite (CPA, templates, regression, MI), and quantitative localization (SLI), enabling in-loop evaluation of countermeasures.

5.3.1. Threat Model and Leakage Observables

We assume a practical, non-invasive adversary with the following capabilities:

- **Measurement:** The attacker has physical access to the device and inserts a small (1–10 Ω) shunt resistor in series with the main V_{DD} supply (Supply Voltage (VDD)). They measure the voltage drop across this shunt to record a single, aggregate power trace, $I_{DD}(t)$.
- **No Internal Access:** The attacker has no internal probes, no access to on-chip clocks, and no dedicated "trigger" signal indicating the start of computation.
- **Alignment:** To overcome the lack of a trigger, we introduce a **virtual trigger**, $V_{\text{trig}}(t)$, which the attacker computes numerically from the *same* power trace. This virtual trigger is used only to segment the trace into spike epochs for analysis.

5.3.2. Substrate-Aware Leakage Model for f-SNNs

Understanding *what* leaks is the first step. Based on the a-IGZO TFT physics and the circuit topology in Figure 5.1, the total supply current $I_{DD}(t)$ can be modeled as the sum of three primary components:

$$I_{DD}(t) \approx I_{bias} + V_{head} \sum_i g_i s_i(t) + \sum_k C_k \frac{dV_k}{dt} + n(t), \tag{5.1}$$

where:

- I_{bias} is the static, idle current of the circuit.
- $V_{head} \sum_i g_i s_i(t)$ is the **quasi-DC offset term**. This is the dominant leakage source. g_i represents synaptic conductances (weights), and $s_i(t)$ is synaptic activity (spike rate). During a spike epoch, active synapses create continuous conduction paths, drawing a sustained current proportional to the total active conductance. This creates an observable DC-level shift in the power trace, consistent with the step-like behavior in Figure 5.2.

- $\sum_k C_k \frac{dV_k}{dt}$ is the transient **displacement current** from charging/discharging internal node capacitances C_k. These currents correspond to sharp spike edges.
- $n(t)$ is device and instrumentation noise.

At the millisecond/kHz scales of SNNs, the quasi-Direct Current (DC) offset term dominates the observable leakage, and it is this feature that FlexSpy primarily targets when building leakage models and attacks.

5.3.3. Trace Synthesis and Spike-Aligned Feature Extraction

Trace Synthesis. The framework uses a Process Design Kit (PDK)-calibrated transient simulation (e.g., in Cadence Spectre) to generate power traces. Neuron and synapse blocks are implemented with a-IGZO TFT models, poly-resistors, and Metal–Insulator–Metal (MIM) capacitors. A compact model of the PDN, including series rail resistance and decoupling capacitors, is crucial for capturing the non-ideal effects of a flexible backplane (e.g., rail droop and coupling between distant cells). Traces are simulated across various process, voltage, and temperature (PVT) corners to emulate realistic manufacturing variation and environmental conditions.

Spike-Aligned Feature Extraction. We apply a three-step process to isolate information-rich features from the raw $I_{DD}(t)$ trace:

1. **Baseline Removal & Filtering:** The idle baseline I_{bias} is estimated from quiescent segments and subtracted. A light low-pass filter suppresses measurement noise while preserving the ms-scale dynamics of the quasi-DC offsets.
2. **Virtual Trigger Alignment:** The attacker computes the virtual trigger $V_{\text{trig}}(t)$ to identify spike epochs. Since the quasi-DC offsets create sharp *edges* at their onset and offset, their derivative is large. The virtual trigger is thus computed as the time-derivative of the supply current (or the shunt voltage):
$$V_{\text{trig}}(t) = \alpha \frac{d}{dt} I_{DD}(t) \equiv \frac{d}{dt} V_{shunt}(t). \tag{5.2}$$
By thresholding $V_{\text{trig}}(t)$, the attacker robustly detects the rising and falling edges of spike windows $[t_0, t_1]$ without any internal trigger.
3. **Per-Window Feature Computation:** For each segmented window w, the framework computes the key leakage feature, the average DC-level shift:
$$\Delta I_{DC}^{(w)} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} (I_{DD}(t) - I_{bias})\, dt, \tag{5.3}$$
which directly measures the quasi-DC term in Equation 5.1. Additional features such as spike count (estimated from V_{trig}), inter-spike intervals (Inter-Spike Interval (ISI)), and coarse timing statistics are also extracted to form a feature vector $Z^{(w)}$ per window.

For downstream attacks we typically concatenate window-wise features into a design-level vector $\mathbf{z}_{\text{design}}$, optionally augmented by device-level indicators (e.g., idle noise, static offsets).

5.3.4. Calibrated Attack Suite

After feature extraction, FlexSpy applies a suite of attacks (summarized in Algorithm 3) to quantify the exploitable information.

Correlation Power Analysis (CPA). CPA is used to localize *when* leakage is strongest. We correlate the feature vector $Z^{(w)}$ (primarily $\Delta I_{DC}^{(w)}$) with a linear predictor of rate-weighted activity,

$$x = \sum_i g_i \bar{s}_i,$$

and compute the Pearson correlation across windows. Peaks in the sliding-CPA trace identify dominant leakage windows and validate the quasi-DC leakage model [12].

Template Profiling (Gaussian). To perform label inference (classifying which input y was processed), we train Gaussian templates (e.g., Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA)) on the feature vectors Z from a profiling set. At attack time, per-window log-likelihoods are summed to form a global score for each candidate label, yielding a powerful profiled attack.

Regression for Continuous Recovery. To recover continuous values such as layer-wise spike rates, we train ridge regressors mapping the feature vector Z to per-layer rate vectors $r^{(l)}$. High R^2, low NMSE, and high cosine similarity indicate that an attacker can accurately estimate intermediate firing rates from power alone.

5.3.5. Leakage Quantification and Localization

Two-Layer Mutual Information Accounting. To distinguish device-specific leakage from design-level leakage, we use a two-layer Mutual Information (MI) decomposition. Let S denote the secret (e.g., label y), L_{device} capture device-specific features (e.g., idle noise, static offsets), and L_{design} denote the spike-window features (e.g., ΔI_{DC}, counts). The total information satisfies

$$I(S; L_{\text{device}}, L_{\text{design}}) = I(S; L_{\text{device}}) + I(S; L_{\text{design}} \mid L_{\text{device}}), \tag{5.4}$$

which we estimate empirically following standard information-theoretic practice [17]. This decomposition reveals how much information stems from the neuromorphic design versus the underlying substrate and fabrication variations.

Spike-Leakage Index (SLI) for Hotspot Ranking. To provide actionable feedback to designers, we must localize *where* and *when* leakage occurs. We define the **Spike-Leakage Index (SLI)** as a normalized MI for a specific circuit block b (e.g., Synapse, Charge, Reset) and time window w:

$$\text{SLI}(b, w; S) = \frac{I(S; Z_b^{(w)})}{H(S)} \in [0, 1], \tag{5.5}$$

where $Z_b^{(w)}$ is the feature vector extracted from block b in window w, and $H(S)$ is the entropy of the secret S. High SLI highlights spatio-temporal hotspots, enabling targeted countermeasures.

5.4. Experimental Evaluation and Results

5.4.1. Experimental Setup

We instantiated the f-SNN and f-RNN circuits in Cadence Virtuoso Spectre using technology-calibrated a-IGZO TFT models. The PDN model included typical parameters for a large-area flexible substrate ($R_{\text{rail}} \approx 6$–$10$ kΩ, $C_{\text{decoupling}} \approx 30$ μF). We simulated millisecond-length traces in the 1–5 kHz regime across nominal, low-V_{DD}, and high-temperature corners.

For workloads, we used time-series datasets from the UCR Archive [40] with a 70/15/15 split for profiling, validation, and attack. Reported metrics include ROC–AUC (ROC-AUC) for label inference, normalized MSE (*NMSE*) and cosine similarity for spike-rate regression, and mutual information (MI) for leakage quantification.

Algorithm 3 FlexSpy End-to-End Procedure

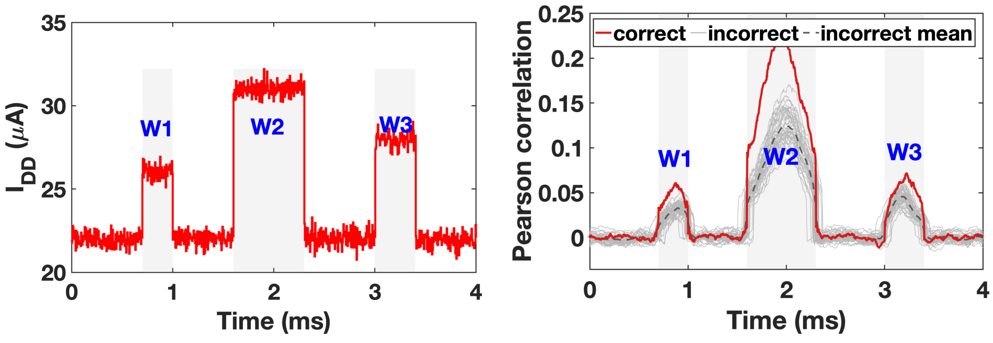
```
1:  Inputs: Netlist $N$, Device Models $M$, Stimuli $X$, Targets $O$, PDN parameters
2:  Outputs: CPA peaks, AUC/Accuracy, $R^2$/NMSE, MI, SLI map, Security–Power trade-offs

3:  Step 1: Synthesize Traces
4:  for all $(x, \text{corner}) \in X \times M$ do
5:      Run transient simulation on $N$ with PDN
6:      Record $I_{DD}(t)$ and compute $V_{\text{trig}}(t) \leftarrow \frac{d}{dt} I_{DD}(t)$

7:  Step 2: Build Features
8:  $I_{DD}(t) \rightarrow$ baseline removal $\rightarrow$ light filtering
9:  Use $V_{\text{trig}}(t)$ to segment spike windows $[t_0, t_1]$
10: Compute per-window features $Z \leftarrow \{\Delta I_{DC}, \text{count}, \text{ISI}, \text{timing}\}$
11: Standardize features by corner

12: Step 3: CPA
13: Correlate windowed $Z$ with hypothesis $x = \sum g_i \bar{s}_i$
14: Record peak correlation and window index $w_{\text{peak}}$

15: Step 4: Profiling (Templates)
16: Fit Gaussian templates (e.g., LDA/QDA) for targets $o \in O$
17: At attack time, sum per-window log-likelihoods for classification

18: Step 5: Regression
19: Train ridge model $Z \mapsto r^{(l)}$ (layer-wise rates)
20: Report $R^2$, NMSE, cosine similarity on held-out traces

21: Step 6: MI / SLI
22: Estimate $I(S; L_{\text{device}})$ and $I(S; L_{\text{design}} \mid L_{\text{device}})$
23: Compute $\text{SLI}(b, w; S) \leftarrow I(S; Z_b^{(w)})/H(S)$ for all $b, w$

24: Step 7: Evaluate Countermeasures
25: Sweep jitter, balancing, and PDN settings
26: Re-run Steps 1–6
27: Report leakage reduction vs. power/latency/area overhead
```

5.4.2. Label Inference from Power Leakage

We first evaluated the ability to infer the input class label y from the spike-window feature vector $\mathbf{z}_{\text{design}}$.

At the nominal corner, a logistic classifier trained on $\mathbf{z}_{\text{design}}$ achieves an ROC–AUC of ROC-AUC $= 0.91$ with low calibration error (ECE $< 5\%$), confirming a strong, class-dependent signal. At low V_{DD} and high temperature, the AUC degrades gracefully to 0.85 and 0.82, respectively. Augmenting the feature vector with simple device-variation indicators (e.g., idle noise) improves AUC by only ≈ 0.02, indicating that class information is dominated by the design-level spike-window features.

Figure 5.4 shows how sliding-CPA localizes the leakage in time. The left panel illustrates the measured supply current $I_{DD}(t)$ with shaded spike windows (W1–W3), where quasi-DC offsets appear during spike epochs in the f-SNN. The right panel plots the sliding-CPA correlation for multiple hypotheses: a bundle of incorrect hypotheses (gray) stays near zero except for small fluctuations, while the correct hypothesis (red) exhibits a sharp correlation peak confined to the dominant spike window W2. This behavior agrees



(a) Measured $I_{DD}(t)$ with spike windows (W1–W3); quasi-DC offsets in f-SNN appear during spike epochs.

(b) Sliding-CPA localization: the correct hypothesis (red) peaks in window W2, incorrect ones (gray) near zero.

Figure 5.4.: Leakage localization in time for f-SNN on P-CONS. Left: quasi-DC ΔI_{DC} offsets in $I_{DD}(t)$ during spike epochs. Right: sliding-CPA shows that leakage is maximized in W2.

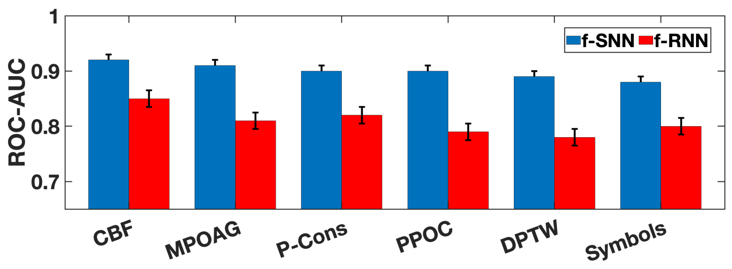


Figure 5.5.: Cross-dataset leakage at the nominal corner: ROC–AUC for label inference using spike-window features on six workloads. Blue bars: f-SNN; red bars: f-RNN; the dotted line indicates chance (AUC = 0.5). The f-SNN consistently leaks more than the f-RNN.

with the quasi-DC leakage model in Equation 5.1 and highlights that ΔI_{DC} drives most of the measurable signal.

Across datasets, the same pattern holds. Figure 5.5 summarizes cross-dataset leakage at the nominal corner: the f-SNN consistently exhibits higher label leakage (higher ROC-AUC) than the f-RNN, with error bars showing 95% confidence intervals across runs. The f-RNN's smoother dynamics and weaker spike alignment yield lower instantaneous leakage, although some information still accumulates over longer time horizons.

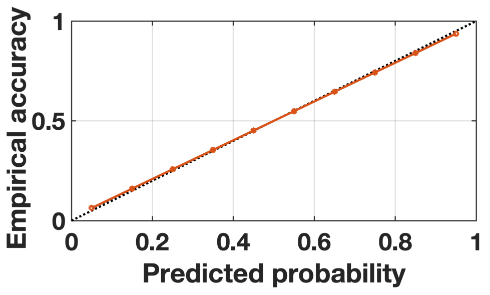
Calibration and robustness to measurement noise are shown in Figure 5.6. The classifier is slightly under-confident but maintains ECE < 5%, and AUC degrades smoothly under added measurement noise, again indicating that spike-window observables dominate over device noise.

Finally, Figure 5.7 visualizes class-dependent leakage for P-CONS. The windowed current shift $\Delta I_{DC}^{(w)}$ in the dominant W2 window differs measurably between classes, directly reflecting the rate-weighted term $V_{head} \sum_i g_i s_i(t)$ in Equation 5.1. This separation explains the high AUC and provides an intuitive picture of the leakage source.

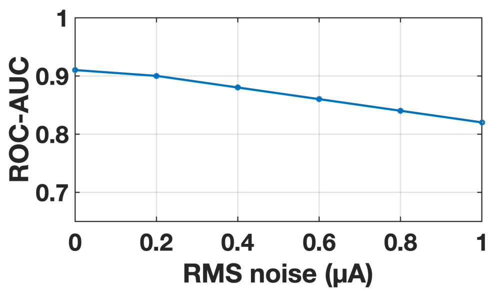
5.4.3. Spike-Rate Recovery and Mutual Information

Beyond classification, we evaluate whether an attacker can reconstruct continuous-valued intermediate data such as layer-wise spike rates. A ridge regressor is trained to map $\mathbf{z}_{\text{design}}$ to the firing-rate vectors $\mathbf{r}^{(\ell)}$ of each layer.

Across PVT corners, we obtain $NMSE = 0.14$ and cosine similarity of 0.93 between predicted and true spike-rate vectors. Figure 5.8 (a) plots predicted vs. true rates on held-out traces, showing a near-linear trend. This linearity reflects the quasi-static term $\sum_i g_i \bar{s}_i$ in Equation 5.1 captured by ΔI_{DC}: average firing rate in each window is essentially encoded in a rate-weighted current offset.



(a) Reliability diagram: the model is slightly under-confident with ECE < 5%, and adding device-variation indicators yields only minor benefit.



(b) AUC vs. added measurement noise (normalized): smooth roll-off, consistent with dominance of spike-window features.

Figure 5.6.: Model reliability and measurement robustness for label inference in f-SNN on P-Cons.

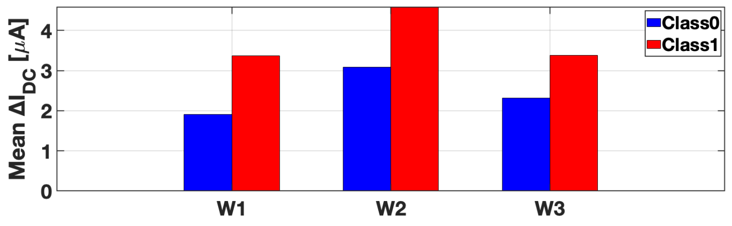


Figure 5.7.: Windowed quasi-DC current shift ΔI_{DC} by class for f-SNN on P-Cons. Distinct class clusters in the dominant W2 window visualize the rate-weighted $\sum_i g_i s_i$ leakage mechanism.

We also apply the two-layer MI decomposition in Equation 5.4. For label leakage, we measure

$$I(y; L_{\text{device}}) \approx 0.09 \text{ bits}, \qquad I(y; L_{\text{design}} \mid L_{\text{device}}) \approx 0.68 \text{ bits},$$

which implies that roughly 88% of class-related information stems from spike-window features, not device-specific fingerprints. For rates and multiplicity, conditional MI lies in the range 0.72–0.81 bits at nominal conditions and drops by $\sim 20\%$ at low V_{DD}. Overall, most recoverable information is attributable to design-level spike windows.

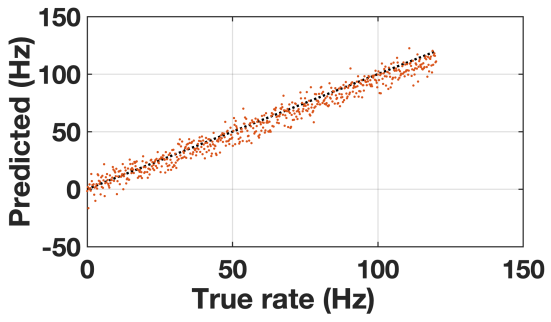
5.4.4. Structural Profiling from Power Traces

We next assess whether power traces leak structural information about the neuromorphic circuit, beyond labels and rates. Gaussian templates are trained on concatenated spike-window features to recover two design-level targets that arise naturally in flexible analog neuromorphic circuits:

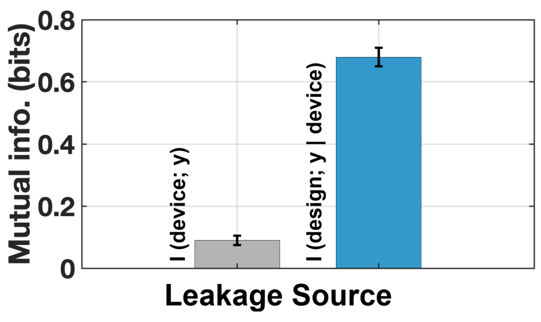
- **Synapse Multiplicity k:** the number of active conductance paths contributing to a spike window ($k \in \{0, \dots, 4\}$ in our experiments).
- **Source Clusters (C1–C4):** groups of synaptic inputs defined by connectivity or receptive-field origin.

At nominal conditions, synapse multiplicity reaches 87.3% accuracy (F1-score 85.9%, ECE 3.2%), with confusions concentrated on adjacent k values, as shown in Figure 5.9 (a). Source-cluster inference (C1–C4) achieves 83.1% Top-1 and 95.2% Top-2 accuracy when incorporating simple timing context (Figure 5.9 (b)). Corner degradation is modest (e.g., 82% / 79% at high-T/low-V_{DD}), confirming the robustness of structural profiling.

These results show that power traces leak not only *what* the network is computing (labels, rates), but also *how* it is structured (multiplicity and connectivity pattern), enabling reverse-engineering of neuromorphic architectures.

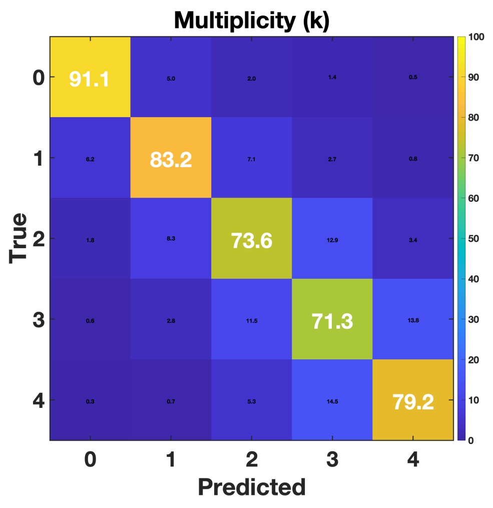


(a) Layer-wise rate recovery on held-out traces (P-Cons); dashed line is identity.

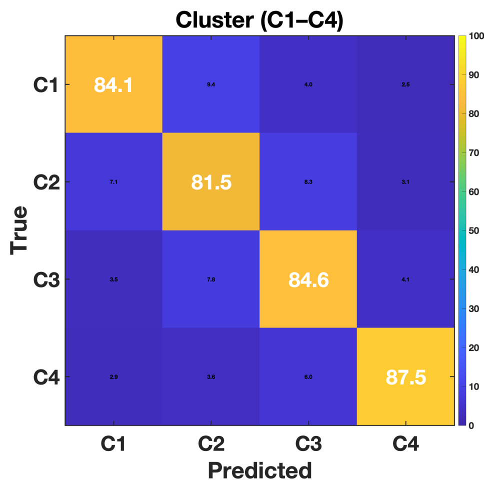


(b) Two-layer MI decomposition: labels leak primarily via design-level spike-window activity.

Figure 5.8.: Spike-rate recovery and mutual information analysis for f-SNN.



(a) Profiling synapse multiplicity k for f-SNN on P-Cons; mistakes are mostly off-by-one.



(b) Source-cluster profiling (C1–C4); errors concentrate on neighboring clusters.

Figure 5.9.: Confusion matrices for structural profiling from power traces in f-SNN. Gaussian templates trained on spike-window features can recover both multiplicity and input source clusters.

5.4.5. Comparative Study: f-SNN vs. f-RNN

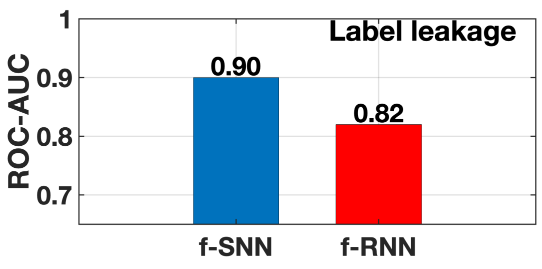
To understand how flexible dynamical primitives differ in leakage behavior, we repeat the analysis for f-RNN circuits that implement second-order *RC* filters. As already hinted by Figure 5.2, the f-RNN produces low-frequency oscillatory envelopes, whereas the f-SNN exhibits quasi-DC steps aligned with spike epochs.

Under identical rails, corners, and workloads, f-SNN achieves higher AUC and lower rate-recovery NMSE due to stronger static-current coupling. Figure 5.10 summarizes the comparison on P-Cons: the f-SNN exhibits stronger label leakage (AUC 0.90 vs. 0.82) and lower NMSE in recovering internal activity (0.14 vs. 0.26). The f-RNN hides instantaneous spikes but accumulates leakage more slowly over time, making it inherently more robust to the spike-window style of attack that FlexSpy targets.

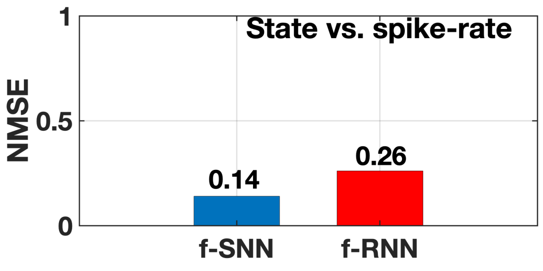
Overall, the comparative study confirms our hypothesis: the event-driven f-SNN is the more vulnerable architecture when leakage is dominated by rate-weighted quasi-DC offsets.

5.5. Countermeasures and Mitigation

A key goal of FlexSpy is to enable *in-loop* evaluation of defenses, allowing designers to iteratively refine circuits at design time. We analyze two lightweight, circuit-level countermeasures that specifically target the quasi-DC leakage mechanism identified in Equation 5.1.



(a) Label leakage (ROC–AUC): f-SNN 0.90 > f-RNN 0.82.



(b) State/rate NMSE: f-SNN 0.14 < f-RNN 0.26.

Figure 5.10.: Direct security comparison of f-SNN vs. f-RNN on P-Cons. The f-SNN's spike-window ΔI_{DC} offsets produce stronger instantaneous leakage than the f-RNN's smoother envelopes.

Table 5.1.: Spike-Leakage Index (SLI) in dominant window W2 for f-SNN on P-Cons. Both countermeasures (CMs) are effective, and the combination provides the strongest hotspot suppression.

Block (W2)	Baseline	+Jitter	+Balancing	+Both
Synapse	0.55	0.43	0.22	**0.17**
Charge/Integrate	0.40	0.31	0.16	**0.12**
Reset/Discharge	0.18	0.14	0.07	**0.05**

5.5.1. Countermeasure 1: Spike-Time Randomization (Jitter)

The first defense aims to break the adversary's ability to align traces. We introduce bounded jitter (typically $\pm 5\%$) into the neuron's reset path, implemented by modulating the effective threshold with a small pseudo-random current source. This jitter desynchronizes spike epochs across measurements, reducing the sharpness of CPA peaks and smearing the ΔI_{DC} features across time. As a result, profiling templates and regression models must average over more variability, weakening both label inference and rate recovery.

5.5.2. Countermeasure 2: Event Balancing (Dummy Conductance)

The second defense directly attenuates the quasi-DC amplitude of ΔI_{DC}. We add a balancing branch (a switched TFT resistor or conductance element) that sources a small, calibrated counter-current whenever a synapse is active. The balancing element is sized to partially cancel the rate-weighted current offset, making the total current draw of an active epoch closer to that of an idle epoch. In feature space, this pulls the class clusters for ΔI_{DC} (cf. Figure 5.7) closer together, reducing separability.

5.5.3. Evaluation of Defenses

We re-run the full FlexSpy pipeline with spike-time jitter and event balancing enabled, both individually and in combination. The SLI metric provides a fine-grained view of hotspot mitigation, while aggregate metrics (AUC, NMSE, MI) quantify global security improvements.

Table 5.1 reports the SLI for the dominant window W2 in the f-SNN on P-Cons. In the unprotected baseline, the Synapse block exhibits an SLI of 0.55 in W2. Enabling only jitter reduces this to 0.43, while balancing alone cuts it to 0.22. The combination of jitter and balancing is most effective, suppressing the primary Synapse leak by approximately 70% to an SLI of 0.17. Charge/Integrate and Reset/Discharge blocks exhibit similar trends, though with smaller absolute SLI values.

Table 5.2 summarizes area/power overhead and leakage reduction across datasets for both f-SNN and f-RNN. For f-SNN, the combined countermeasures reduce total leakage by 50–70% with area overheads of 3.5–5.0%, power overheads of 7.0–9.0%, and accuracy impact below 0.5 percentage points. The f-RNN, which exhibits weaker baseline leakage, still benefits from 22–38% leakage reduction at slightly lower overheads.

Table 5.2.: Per-dataset countermeasure (CM) overheads and leakage reduction, grouped by architecture. Values are relative to each model's unprotected baseline; medians across parameter sweeps are reported.

Dataset	f-SNN				f-RNN			
	Area↑ (%)	**Power↑ (%)**	**Acc. Δ (pp (%))**	**Leak↓ (%)**	**Area↑ (%)**	**Power↑ (%)**	**Acc. Δ (pp (%))**	**Leak↓ (%)**
CBF	3.5	7.0	−0.2	60	2.8	5.5	−0.2	29
P-Cons	4.0	8.0	−0.3	56	3.0	6.0	−0.2	24
PPOC	4.5	8.5	−0.4	53	3.2	6.2	−0.3	23
DPTW	5.0	9.0	−0.5	50	3.5	6.5	−0.3	22
MPOAG	4.0	8.0	−0.3	55	3.0	6.0	−0.2	25
Symbols	3.5	7.0	−0.2	70	2.8	5.5	−0.2	38

CM = spike-time jitter (±5%) + event balancing. f-SNN benefits more because its leakage is dominated by spike-window ΔI_{DC} offsets, while f-RNN has smoother, lower-contrast envelopes.

Overall, the countermeasures offer a highly favorable security–power trade-off. FlexSpy thus not only diagnoses f-SNN vulnerabilities but also validates practical, substrate-aware defenses.

5.6. Summary

This chapter introduced FlexSpy, the first substrate-aware, design-time security framework for flexible spiking neuromorphic hardware. We demonstrated that the unique combination of flexible substrates (thin, unshielded, shared rails) and SNN computational dynamics (event-driven, sparse) creates a significant and previously unquantified power side-channel vulnerability.

The core technical contribution is a unified device-to-design-to-network model that links the physics of a-IGZO TFTs to observable, network-level information leakage. We identified the dominant leakage primitive: a rate-weighted, quasi-DC current offset stemming from active synaptic conductances during spike epochs. Using only a single, non-invasive power measurement and a self-generated virtual trigger, an attacker can exploit this primitive to infer labels with ROC-AUC = 0.91 and recover layer-wise spike rates with $NMSE = 0.14$.

Our comparative analysis showed that the f-SNN's event-driven model is substantially more "leaky" than the smoother dynamics of an f-RNN, both in terms of label inference and internal state reconstruction. To provide actionable design guidance, we developed the Spike-Leakage Index (Spike-Leakage Index (SLI)), an MI-based metric that successfully localizes security hotspots in time and across circuit blocks.

Finally, we demonstrated that this vulnerability is not fundamental. Two lightweight countermeasures, spike-time jitter and event balancing, were proposed and validated. Together, they suppress quasi-DC leakage by 38–70% with modest power overhead ($\leq$ 9%), small area cost, and negligible accuracy degradation. FlexSpy thus provides a necessary, early-stage assessment tool and a pathway towards building a new generation of flexible neuromorphic devices that are *secure by design*. Future work will extend this framework to other channels (e.g., EM emissions) and explore co-design of training algorithms and circuit primitives to further harden f-SNNs against side-channel threats.

6. Vulnerability of Edge-AI-based Hyperdimensional Computing: FPGA Emulation

6.1. Introduction

The rapid proliferation of AI at the network edge has fundamentally altered the architectural requirements of modern computing systems. As applications such as autonomous driving, wearable health monitoring, and industrial IoT expand, the constraints of power, latency, and bandwidth have rendered cloud-centric DNNs are increasingly untenable for real-time, localized processing.

HDC has emerged as a transformative paradigm in this context. By mimicking the high-dimensional, distributed representation of information found in biological neural circuits, HDC offers a lightweight, energy-efficient alternative to traditional Von Neumann architectures. Its reliance on simple, highly parallelizable bitwise operations, such as XOR and population counts, makes it an ideal candidate for hardware acceleration on FPGAs.

However, the transition from algorithmic theory to physical implementation introduces a critical, often overlooked dimension: hardware security. While HDC is celebrated for its intrinsic algorithmic robustness, its ability to tolerate stochastic noise and random bit flips without significant degradation in inference accuracy, this characteristic creates a dangerous *robustness paradox*. The very redundancy that protects HDC from random noise can obfuscate the presence of targeted malicious attacks, while the highly optimized, parallel hardware structures required for efficiency inadvertently expose broad attack surfaces.

This chapter provides an in-depth analysis of the physical security posture of FPGA-based HDC accelerators. Synthesizing empirical data from three recent works, [108], [105], and [106], we construct a holistic threat model that spans passive SCA, remote internal sensing, and active fault injection (FI). Through the examination of over one million power traces and cross-platform validation on Artix-7, Arty-A7, and Zynq-based architectures, we show that unprotected HDC accelerators are vulnerable to:

- **Intellectual Property (IP) theft**: model extraction rates exceeding 90% via bit-level recovery of class hypervectors (ClassHVs);
- **Integrity violations**: targeted misclassification rates approaching 90% under carefully injected faults.

We further evaluate the efficacy of several countermeasures, including Adaptive Grad-CAM-guided defenses, dynamic masking, and hypervector randomization, demonstrating that security can be substantially improved with modest resource overhead.

6.2. Theoretical Framework: Hyperdimensional Computing and Hardware Vulnerability

6.2.1. The Algebra of Hypervectors and Hardware Mapping

HDC represents information as *hypervectors*, pseudo-random vectors of high dimensionality, typically $D \geq 1{,}000$ bits. Rather than operating on precise numerical values, HDC manipulates distributed binary patterns. The core algebra is built around three primary operations:

Table 6.1.: Mapping of core HDC operations to FPGA resources and associated security implications.

HDC Operation	Math Function	FPGA Mapping	Security Implication
Binding	$\mathbf{A} \oplus \mathbf{B}$	Parallel XOR gates (LUTs)	High switching activity; leakage correlates with Hamming distance.
Bundling	$\mathrm{sgn}(\mathbf{A} + \mathbf{B} + \dots)$	Adder trees and comparators	Complex power profile; prone to overflow/saturation faults.
Similarity	$\sum(\mathbf{A} \oplus \mathbf{B})$	POPCOUNT logic and accumulators	Primary leakage point; power linearly tracks hypervector similarity.
Storage	Memory read/write	Block RAM / distributed RAM	Susceptible to read/write fault injection; read accesses leak ClassHV contents.

1. **Binding (Association)**: associates two pieces of information (e.g., a symbol and its value). Mathematically, binding is often realized via dimension-wise Exclusive-OR,

$$\mathbf{C} = \mathbf{A} \oplus \mathbf{B},$$

producing a third hypervector that is nearly orthogonal to both inputs.

2. **Bundling (Superposition)**: aggregates multiple hypervectors into a single representative hypervector, typically via component-wise addition followed by a thresholding or majority function,

$$\mathbf{C} = \mathrm{sgn}\left(\sum_j \mathbf{A}_j\right),$$

where the sign/threshold operator is applied dimension-wise.

3. **Permutation (Sequence Encoding)**: encodes order (e.g., positions in a sequence or time series) by permuting coordinates of a hypervector, such as a rotation or cyclic shift operation.

The inference phase, the primary target of side-channel and fault attacks, involves calculating similarity between a query hypervector (QueryHV) and stored class hypervectors (ClassHVs). The standard similarity metric is a Hamming distance (or equivalently, Hamming similarity), computed via XOR followed by a POPCOUNT (population count).

These abstract operations map to specific FPGA resources and carry distinct security implications, summarized in Table 6.1.

As Table 6.1 highlights, similarity computation (XOR + POPCOUNT) is the most critical from a side-channel perspective: power consumption is directly correlated with the Hamming distance between QueryHV and ClassHV, providing an exploitable leakage source.

6.2.2. Hyperdimensional Computing Hardware

HDC, as a machine-learning classifier, represents complex raw data as high-dimensional hypervectors. Two fundamental operations, *binding* and *bundling*, are used to encode structure and aggregate information. Data elements are represented as randomly initialized hypervectors, referred to as *item vectors*. For instance, encoding a binary image involves generating an *item memory* containing a unique hypervector for each pixel position. Hypervectors corresponding to active pixels are then bundled into a single hypervector representing the image [46], [52].

Figure 6.1 shows the architecture of the FPGA-based HDC accelerator considered in this chapter. During training, hypervectors from samples of the same class are aggregated into representative *ClassHV*s, stored in an associative memory (AM). During inference, input samples undergo the same encoding, producing *QueryHV*s [73], [75], [90], [91]. Classification is performed by comparing *QueryHV*s with

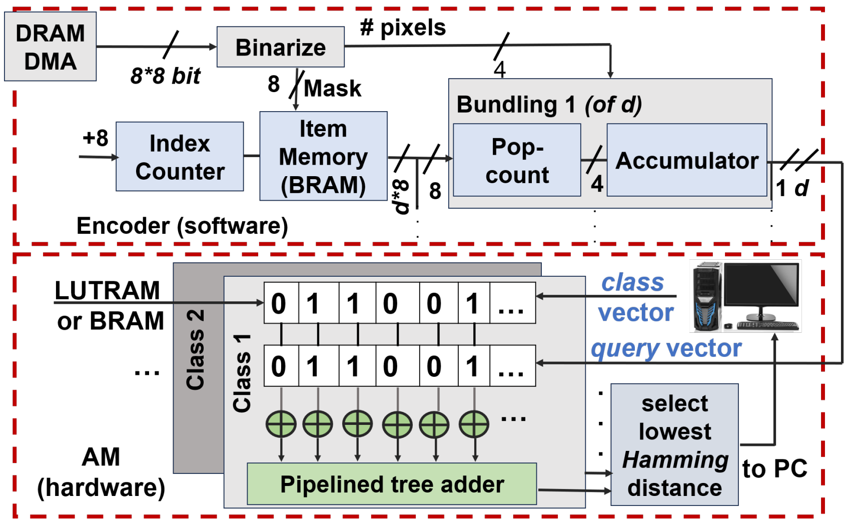


Figure 6.1.: Architecture of the FPGA-based HDC accelerator. Raw data is streamed via DMA, encoded as hypervectors, and classified in the associative memory (AM). Class hypervectors are written by the CPU and stored in AM (large hypervectors are partitioned into segments). The class with the smallest Hamming distance to the query is returned as the classification result.

stored *ClassHV*s using similarity metrics (Hamming distance / POPCOUNT) realized in the AM. Our HDC configuration reflects workloads similar to sensor classification tasks in edge devices, such as gesture recognition and health monitoring.

6.2.3. The Threat Landscape: Physical Access and Adversarial Capability

We adopt three different threat models tailored to edge deployments for three different works:

- **Passive power profiling (local SCA):** The adversary attaches external equipment (e.g., ChipWhisperer Pro, oscilloscope) to measure the power consumption of the FPGA while the HDC accelerator performs inference. The goal is *confidentiality compromise*: extracting ClassHVs and thereby stealing the trained model IP.
- **Dynamic workload scheduling:** The adversary is on the environment of dynamic workload scheduling to efficiently handle multiple AI inference tasks sequentially using static FPGA resources [99](popular for edge-AI devices). Unlike cloud-based FPGA setups, where multiple workloads are explicitly partitioned and simultaneously executed [83], [92], dynamically scheduled edge FPGAs lack explicit synchronization signals, fundamentally changing how side-channel leakage can be observed.
- **Active fault injection FI:** The adversary manipulates supply voltage, clock, or environmental conditions to induce timing violations or logic upsets. The goal is *integrity compromise*: forcing targeted misclassifications or degrading reliability by corrupting similarity computations or memory contents.

In the remainder of this chapter we first focus on the local power side-channel attack vector, demonstrating that deep-learning-based analysis can recover sensitive bits with high accuracy. Subsequent sections build upon this to analyze dynamic workload scheduling scenarios and fault-based attacks.

6.3. Deep Learning-Assisted Power Attack

6.3.1. The Dimensionality Limitations in Classical SCA

Classical (SCA) techniques such as Correlation Power Analysis (CPA) and Differential Power Analysis (DPA) assumes that at a given time instant, power consumption is linearly correlated with the Hamming

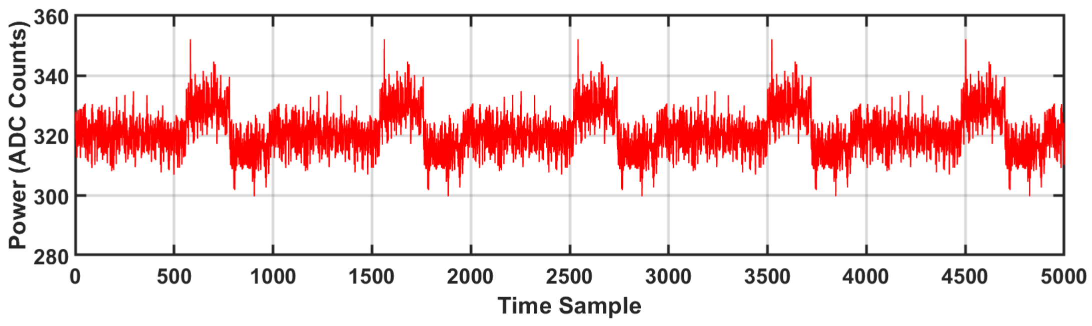


Figure 6.2.: *ChipWhisperer*-captured FPGA power trace showing different stages of HDC operations. The trace exhibits a periodic pattern after a fixed timestamp, corresponding to similarity computations, which is critical for SCA.

weight or Hamming distance of the secret-dependent data being processed. This assumption holds reasonably well for cryptographic primitives and small datapaths, but it breaks down in HDC accelerators.

HDC operations are inherently parallel: a single clock cycle may involve the switching of thousands of bits (e.g., $D = 5{,}000$–$10{,}000$). The switching activity associated with any single secret bit is thus superimposed on the aggregate activity of the remaining $D - 1$ bits, creating a holographic noise effect. From an SCA perspective, the signal-to-noise ratio (SNR) of any individual bit is extremely low.

Empirically, standard difference-of-means or correlation-based tests fail to reliably distinguish the leakage of a single target bit, even when millions of traces are available. The attacker would require an impractically large number of measurements to overcome the dimensionality-induced noise floor. This motivates the use of deep learning to automatically learn nonlinear, high-dimensional leakage patterns from raw traces.

Figure 6.2 shows a representative power trace captured from the FPGA-based HDC accelerator using ChipWhisperer. Different HDC pipeline stages produce distinct patterns, and a periodic structure emerges after a fixed timestamp corresponding to repeated similarity computations. Identifying and exploiting these regions is critical for successful SCA.

6.3.2. Attack Methodology

To overcome the high-dimensional noise floor, we adopt a deep-learning-based SCA (DL-SCA) approach inspired by recent work on HDC leakage. The method employs a specialized 1-dimensional Residual Neural Network (ResNet-34) to operate directly on raw power traces, without manual point-of-interest selection.

The choice of a deep residual architecture is strategic:

- **Hierarchical feature extraction:** Unlike template attacks that rely on manually selected time samples, the ResNet-34 learns to automatically extract features from the raw trace. Early layers capture low-level switching transients, while deeper layers encode higher-level patterns associated with bundling, binding, and similarity operations.

- **Mitigation of vanishing gradients:** Residual connections facilitate gradient flow through many layers, enabling efficient training on long, noisy time series such as power traces. This is crucial when traces span many clock cycles and contain correlated noise.

6.3.2.1. Adaptive Grad-CAM-Guided Attack Framework

A key innovation is the integration of Gradient-weighted Class Activation Mapping (Grad-CAM) into an *adaptive* attack loop. Grad-CAM was originally developed for computer vision to visualize which spatial regions of an image influence a CNN's decision. Here, we repurpose it for temporal leakage localization.

At a high level, Grad-CAM is used to compute an importance score for each time sample in a trace, indicating how much that sample influences the CNN's ability to predict a target bit. Figure 6.3 illustrates

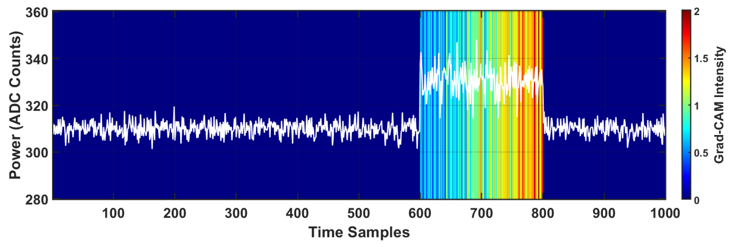


Figure 6.3.: Grad-CAM heatmap highlighting regions within FPGA-based HDC power traces that contribute most to ResNet-34 predictions. The highlighted intervals correspond primarily to XOR similarity and POPCOUNT computations.

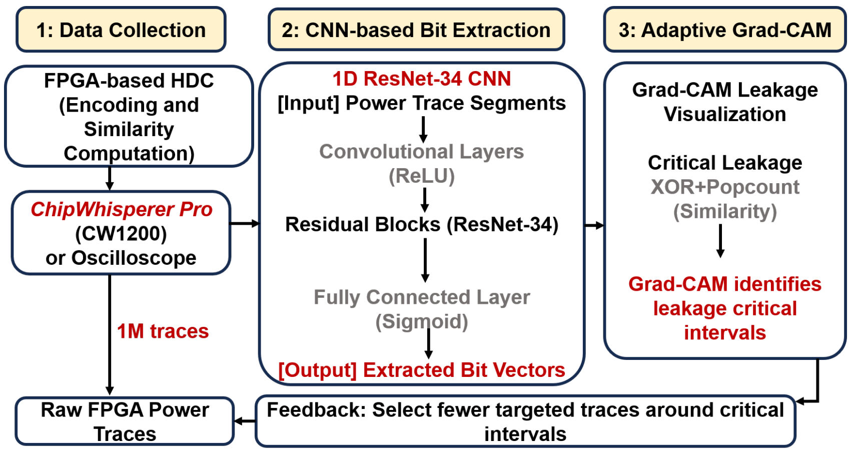

1: Data Collection
2: CNN-based Bit Extraction
3: Adaptive Grad-CAM
FPGA-based HDC (Encoding and Similarity Computation)
ChipWhisperer Pro (CW1200) or Oscilloscope
1M traces
Raw FPGA Power Traces
1D ResNet-34 CNN
[Input] Power Trace Segments
Convolutional Layers (ReLU)
Residual Blocks (ResNet-34)
Fully Connected Layer (Sigmoid)
[Output] Extracted Bit Vectors
Grad-CAM Leakage Visualization
Critical Leakage XOR+Popcount (Similarity)
Grad-CAM identifies leakage critical intervals
Feedback: Select fewer targeted traces around critical intervals

Figure 6.4.: Overview of the proposed adaptive Grad-CAM-based attack workflow. Power traces from HDC inference are fed to a 1D ResNet-34 CNN for bit extraction. Grad-CAM visualizations identify and refine critical leakage intervals, guiding targeted trace selection and cropping to enhance attack efficiency.

Grad-CAM heatmaps highlighting the temporal regions where XOR and POPCOUNT logic contribute the most to leakage.

The adaptive attack proceeds iteratively, as summarized in Figure 6.4 and Algorithm 4:

1. **Initial training:** Train the ResNet-34 on a broad set of traces covering the entire inference operation, without prior alignment or cropping.
2. **Leakage visualization:** Apply Grad-CAM to the trained model to obtain a heatmap over time. Each time point receives a score indicating its contribution to correctly predicting the secret bit.
3. **Critical interval identification:** Identify high-score intervals, or “hotspots”, which correspond to clock cycles where XOR similarity and POPCOUNT accumulate are performed.
4. **Targeted refinement:** Collect or select new traces and crop them to focus on these intervals, effectively filtering out irrelevant pipeline stages (e.g., encoding or DMA transfers) and thereby improving SNR.
5. **Retraining:** Retrain the network on these refined traces, iterating until bit extraction accuracy saturates.

6.3.3. Empirical Evaluation and Transferability

We validate the adaptive DL-SCA methodology on a real-hardware setup using a ChipWhisperer Pro (CW1200) attacking a dedicated Artix-7 FPGA target board (CW305). Approximately one million power traces are collected during HDC inference, covering multiple benchmark datasets (MNIST, ISOLET,

Algorithm 4 Adaptive Grad-CAM-Guided 1D ResNet-34 SCA

```
Require: Initial dataset $X_{\text{init}}$, labels (bits) $Y$, iterations $T$, initial model $f_\theta^{(0)}$
Ensure: Optimized model $f_\theta^{(T)}$, minimized trace count
1: for $t = 1$ to $T$ do
2:     Train CNN: $f_\theta^{(t)} \leftarrow \text{train}(f_\theta^{(t-1)}, X^{(t-1)}, Y)$
3:     Apply Grad-CAM to identify critical leakage intervals:
4:       $I_{\text{critical}}^{(t)} \leftarrow \text{GradCAM}(f_\theta^{(t)}, X^{(t-1)}, Y)$
5:     Refine data collection: select new traces around critical intervals:
6:       $X^{(t)} \leftarrow \text{SelectTraces}(I_{\text{critical}}^{(t)}, X_{\text{available}})$
7:     Evaluate bit extraction accuracy on a validation set
8:     if accuracy is sufficiently high then terminate early
9: return $f_\theta^{(T)}$
```

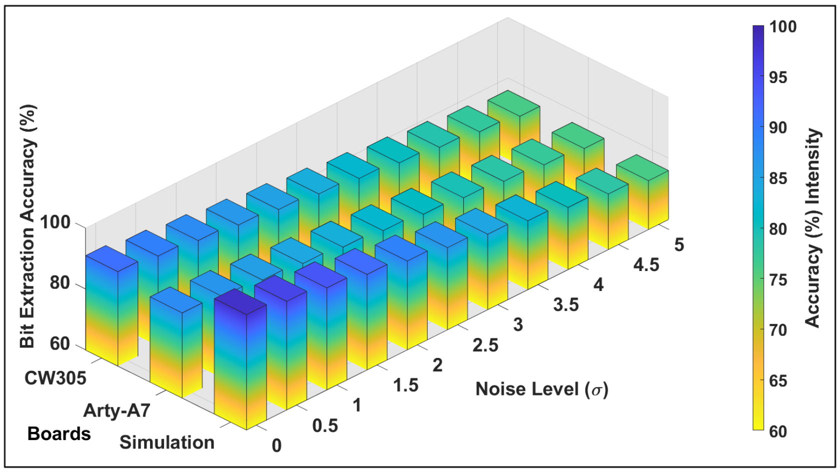


Figure 6.5.: ResNet-34-based bit extraction accuracy across FPGA platforms under explicit noise and jitter conditions. Cross-platform consistency highlights that leakage is dominated by architectural similarity (XOR + POPCOUNT structure) rather than chip-specific artifacts.

HAR, CARDIO, CIFAR-10). For each dataset, traces are captured under a profiling attack scenario, with labels corresponding to individual bits of selected ClassHVs.

High-level observations:

- The adaptive Grad-CAM-guided ResNet-34 achieves high bit extraction accuracy (up to $\approx 93\%$) while substantially reducing the number of traces needed to surpass practical thresholds (e.g., 85% accuracy).
- The learned leakage features are *transferable* across FPGAs. Models trained on CW305 traces can be applied to Arty-A7 traces (Tektronix MSO2024B), with modest loss of accuracy, demonstrating that leakage is largely architectural rather than chip-specific.
- Dataset complexity plays a role: tasks with more complex and higher-entropy inputs (e.g., CIFAR-10) yield slightly lower bit extraction accuracy than simpler tasks such as MNIST or voice/gesture recognition, but remain well above the level required for effective model reconstruction.

Figure 6.5 illustrates how bit extraction accuracy varies across FPGA platforms and noise conditions, motivating a more detailed, quantitative evaluation of SCA effectiveness in the following subsection.

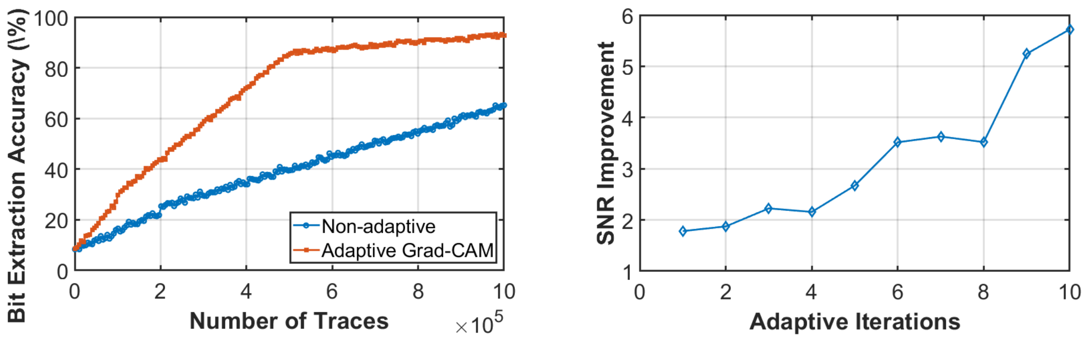


Figure 6.6.: Impact of adaptive Grad-CAM guidance. Left: bit extraction accuracy improvement across iterations for adaptive vs. non-adaptive approaches. Right: SNR improvement across adaptive iterations. The adaptive method achieves significantly higher accuracy with fewer traces by focusing on critical leakage intervals.

Table 6.2.: Detailed performance analysis of CNN variants for FPGA-based HDC side-channel attack (MNIST dataset).

CNN Model	# Residual Blocks	Bit Extraction Accuracy (%)	Inference Speed (queries/s)	Training Time (GPU-hours)	Min. Traces for >85% Accuracy
CNN (Base)	–	81.3	420	6.8	–
ResNet-20	10	89.1	360	8.4	~ 0.95M
ResNet-34	**16**	**93.8**	340	9.6	~ 0.5**M**

6.3.4. Evaluation of Side-Channel Attack Effectiveness

Building on the experimental setup described above, we now quantify the effectiveness of the proposed DL-SCA across CNN architectures, datasets, FPGA platforms, and measurement conditions. We emphasize both bit extraction accuracy and robustness under realistic noise, jitter, and masking countermeasures.

6.3.4.1. Performance Analysis of CNN Variants

We first compare the adaptive Grad-CAM-guided approach across three CNN architectures: a standard 1D CNN, a 1D ResNet-20, and the proposed 1D ResNet-34. Figure 6.6 shows how accuracy and SNR evolve across Grad-CAM iterations.

The adaptive CNN-based method improves both accuracy and data efficiency:

- It attains up to 93% bit extraction accuracy, nearly doubling the accuracy of a non-adaptive baseline using the same number of traces.
- It provides $\approx 1.7\times$ higher *maximum* bit extraction accuracy than the non-adaptive strategy, particularly when using ResNet-34.

Table 6.2 summarizes bit extraction accuracy, inference speed, training time, and minimum trace count required to exceed 85% accuracy for each CNN variant on MNIST traces.

Despite its slightly higher training time and modestly lower inference throughput, the 1D ResNet-34 provides the best trade-off for SCA: it captures subtle, high-dimensional leakage patterns more effectively than shallower networks.

6.3.4.2. ResNet-34-Based Bit Extraction Accuracy

We next evaluate ResNet-34 across multiple datasets and measurement configurations. Table 6.3 summarizes bit extraction accuracy, required traces, training time, and inference speed for three categories:

1. **ChipWhisperer CW1200 measurements (CW305 FPGA)**: baseline, high-quality power traces.
2. **Oscilloscope measurements (Arty-A7 FPGA)**: more noise and jitter, representative of less specialized lab setups.

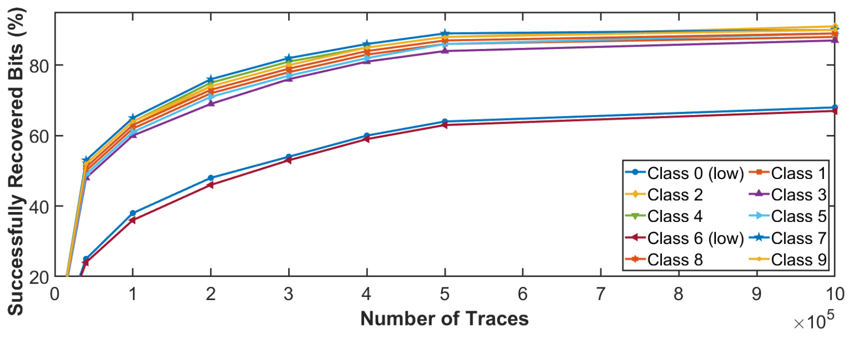


Figure 6.7.: Bit recovery accuracy vs. number of analyzed traces for MNIST. ResNet-34-based SCA converges above 90% accuracy for most classes after $\approx 8 \times 10^5$ traces; some classes (0 and 6) remain harder to recover due to weaker leakage.

3. **Vivado post-implementation power simulation (Zynq-7000)**: noise-free, idealized environment useful for upper-bound analysis.

Table 6.3.: Summary of ResNet-34-based SCA evaluation across multiple benchmark datasets and FPGAs (1M trace budget unless noted).

Dataset	FPGA Platform	Trace Collection Method	Bit Extraction (%)	Traces Required	Training Time (h)	Inference Speed (queries/s)	Remarks
ChipWhisperer CW1200 Measurement (CW305 FPGA)							
MNIST	CW305	CW1200	93	800,000	9.2	420	Baseline; consistently high leakage
ISOLET	CW305	CW1200	91	750,000	7.2	410	Stable leakage characterization
HAR	CW305	CW1200	90	780,000	7.0	412	Slightly reduced accuracy due to data complexity
CARDIO	CW305	CW1200	89	830,000	7.5	405	Moderate noise sensitivity
CIFAR-10	CW305	CW1200	88	880,000	8.0	395	Lower accuracy due to more complex data
Oscilloscope Measurement (Arty-A7 FPGA)							
MNIST	Arty-A7	Tektronix MSO2024B	91	850,000	8.2	395	Measurement noise impact evident
ISOLET	Arty-A7	Tektronix MSO2024B	88	820,000	7.9	390	Noise slightly reduces accuracy
HAR	Arty-A7	Tektronix MSO2024B	87	840,000	8.0	388	Consistent under realistic noise
CARDIO	Arty-A7	Tektronix MSO2024B	86	890,000	8.5	380	Elevated noise sensitivity
CIFAR-10	Arty-A7	Tektronix MSO2024B	84	920,000	9.0	375	Accuracy impacted by high jitter
Vivado Post-Implementation Power Simulation (Zynq-7000)							
MNIST	Zynq-7000	Vivado Simulation	98	500,000	6.0	440	Idealized, noise-free baseline
ISOLET	Zynq-7000	Vivado Simulation	96	500,000	6.5	435	High baseline without practical artifacts
HAR	Zynq-7000	Vivado Simulation	95	520,000	6.8	430	Consistent extraction accuracy
CARDIO	Zynq-7000	Vivado Simulation	95	550,000	7.1	425	Strong accuracy in simulated environment
CIFAR-10	Zynq-7000	Vivado Simulation	93	580,000	7.4	420	Slight reduction from complex patterns
Masked FPGA Implementations (Dynamic Masking Countermeasure)							
MNIST	CW305	CW1200	18	800,000	7.5	360	Significant leakage reduction verified
ISOLET	CW305	CW1200	21	750,000	7.9	355	Consistent masking effectiveness
HAR	CW305	CW1200	18	780,000	8.0	352	Effective leakage mitigation
CARDIO	CW305	CW1200	17	830,000	8.3	348	Reliable protection against leakage
CIFAR-10	CW305	CW1200	19	880,000	8.7	345	High protection despite data complexity

As expected, simulated traces provide an upper bound on bit extraction accuracy, but real measurements on CW305 still achieve 88–93% accuracy across datasets. Arty-A7 traces, which are noisier and less controlled, exhibit modest accuracy degradation but remain clearly above the threshold needed for effective model extraction. Masked implementations, discussed later as a countermeasure, dramatically reduce bit extraction accuracy to 17–21%, near the random guess baseline.

Figure 6.7 shows how bit recovery accuracy improves as the number of analyzed traces increases for MNIST. Accuracy starts at $\approx$ 40–50% for fewer than 2×10^5 traces, then rapidly improves and stabilizes beyond $\approx 8 \times 10^5$ traces, exceeding 90% for most classes. Classes 0 and 6 converge at lower accuracy ($\approx$ 60%), indicating weaker or less distinguishable leakage patterns for those classes (e.g., lower switching activity).

6.3.4.3. Higher-Order Leakage Analysis (TVLA)

Grad-CAM analysis in subsection 6.3.2 and subsubsection 6.3.2.1 identified XOR and POPCOUNT operations as dominant leakage sources. To systematically quantify leakage order and severity, we perform Test Vector Leakage Assessment (TVLA) on unprotected HDC implementations.

Figure 6.8 reports first- and second-order TVLA statistics for MNIST on CW305. Significant first-order leakage is observed (t-values > 4.5) during similarity computation, and notable second-order leakage

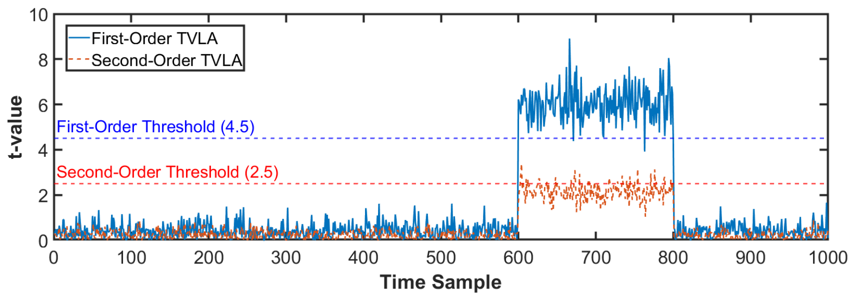


Figure 6.8.: TVLA analysis on unprotected HDC (MNIST): severe first-order leakage (t-values > 4.5) and non-negligible second-order leakage (t-values > 2.5) during XOR and POPCOUNT operations, confirming vulnerability to both first- and higher-order SCA.

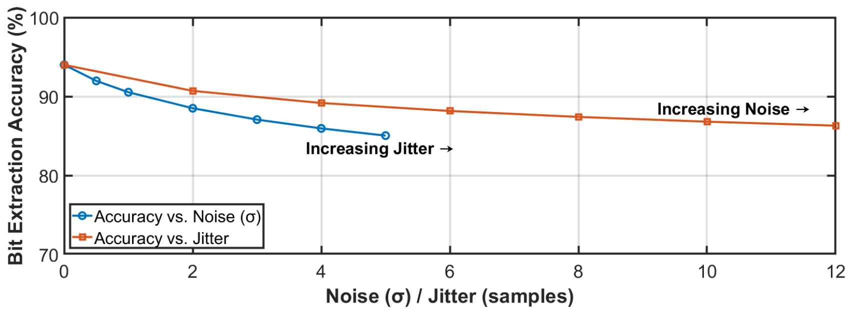


Figure 6.9.: Robustness analysis (MNIST). Bit extraction accuracy of ResNet-34 under increasing Gaussian noise and timing jitter. While accuracy degrades from ideal conditions, the attack remains effective across realistic noise/jitter levels.

(t-values > 2.5) arises from nonlinear interactions and glitches. These results confirm that the unprotected HDC accelerator is highly vulnerable not only to basic first-order attacks but also to higher-order SCA.

6.3.4.4. Impact of Noise and Jitter on CNN Attack Robustness

Real-world traces inherently contain measurement noise and timing jitter. While CW1200 provides high-quality captures, other platforms (or field deployments) may exhibit more severe imperfections. To systematically characterize the impact of such imperfections on CNN-based SCA, we generate synthetic traces using Vivado post-implementation simulations and inject controlled Gaussian noise and timing jitter.

Figure 6.9 shows ResNet-34 bit extraction accuracy as a function of incrementally introduced noise and jitter for MNIST. In the ideal case (no noise, no jitter), accuracy exceeds 98%. Introducing realistic Gaussian noise slightly reduces accuracy; adding timing jitter further degrades performance to $\approx$ 84–87%. Nonetheless, the attack remains effective across a wide range of conditions, emphasizing the need to consider SCA robustness under realistic, noisy environments.

6.3.4.5. Comparison with State-of-the-Art Attacks

Finally, Table 6.4 compares the proposed ResNet-34-based SCA with representative state-of-the-art attacks on FPGA-HDC accelerators. Prior works have demonstrated:

- Model inversion and reconstruction of classifier decision boundaries;
- Adversarial parameter perturbations (e.g., RowHammer-like bit flips) to induce misclassifications;
- Voltage and thermal stress to inject faults into HDC implementations.

Our DL-SCA attack is complementary: rather than perturbing the model, it passively extracts ClassHV bits with high accuracy, enabling IP theft and subsequent off-line cloning of HDC models. Combined with prior FI techniques, this creates a rich attack surface for both confidentiality and integrity violations.

Table 6.4.: Comparative analysis of ResNet-34-based SCA with representative state-of-the-art attacks on FPGA-HDC.

Aspect	**Hernandez et al. [84]**	**Liu et al. [94]**	**Krautter et al. [93]**	**This Work**
Attack Type	Model inversion	Adversarial parameter	Voltage/thermal stress	Power side-channel
Target Platform	FPGA-HDC	FPGA-HDC	FPGA-HDC	FPGA-HDC
Attack Method	Feature reconstruction	Bit flipping (RowHammer-style)	Voltage/thermal variations	CNN-based trace analysis
Vulnerability Exploited	Associative memory	Associative memory	Implementation logic	Similarity computation leakage
Key Outcome	High reconstruction accuracy	Targeted misclassification ($\sim 90\%$)	Significant accuracy degradation	Bit recovery up to $\sim 93\%$
Trigger Mechanism	Reconstruction queries	Parameter perturbation	Stress conditions	Profiling-based power traces
Countermeasures Evaluated	Noise injection, quantization	–	–	Dynamic masking, randomization
Evaluation Metrics	Information leakage	Attack success rate	Fault-induced accuracy drop	Bit recovery, TVLA, robustness

6.4. Dynamic workload scheduling and Internal Sensing SCA Attack: Collision-Based Timing Analysis

While the CNN-based power SCA discussed above requires external probing of the FPGA power supply, a more insidious threat arises when an attacker can deploy logic within the same FPGA fabric as the victim HDC accelerator. This scenario is common in multi-tenant cloud FPGAs or SoC-based edge nodes where user designs share on-chip resources.

Edge platforms, particularly FPGA-based devices such as *AMD-Xilinx Kria* System-on-Module (SoM)**kalapothas2022efficient** and *Efinix Titanium* FPGAs**czymmek2023review**, frequently employ dynamic workload scheduling to efficiently handle multiple AI inference tasks sequentially using static FPGA resources. Unlike cloud-based FPGA setups, where multiple workloads are explicitly partitioned and simultaneously executed, dynamically scheduled edge FPGAs lack explicit synchronization signals, fundamentally changing how side-channel leakage can be observed. Consequently, traditional remote side-channel attacks commonly applied in cloud FPGA setups cannot be directly applied to edge FPGA deployments. This introduces unique challenges, motivating new approaches tailored to the characteristics of dynamically scheduled HDC-based edge environments.

In such environment, the adversary may not have physical access to supply rails or pins, but can still monitor shared on-chip infrastructure. A powerful primitive is to instantiate on-chip time-to-digital converters (TDCs) or ring-oscillator sensors that sense voltage fluctuations on the shared PDN. By carefully timing their own switching activity to *collide* with the victim's HDC operations, the attacker can perform collision-based timing and power analysis without any external measurement equipment.

6.4.1. Attack Overview and Threat Model

We consider an attacker whose primary goal is to compromise the confidentiality of stored class hypervectors (ClassHVs) in an FPGA-based HDC accelerator. These ClassHVs represent the trained model's intellectual property and are sufficient to replicate or clone the model. Given HDC's robustness to noise [47], even partial recovery of ClassHV bits (e.g., 70–80%) is enough to build a highly accurate replica.

Our threat model assumes:

1. **Encoding knowledge:** The attacker knows the hypervector encoding scheme (dimension D, bundling, binding, and permutation rules) and can generate structurally valid query hypervectors (QueryHVs).

2. **Query access:** The attacker can submit controlled inputs (QueryHVs or raw data) to the HDC classifier and observe the predicted label or confidence information via the normal system interface.

3. **On-chip sensor placement:** The attacker can synthesize a small sensor core (TDC) into the same FPGA, sharing the PDN with the victim HDC accelerator. This is realistic in partially reconfigurable or dynamic workload spaces, similar to a multi-tenant cloud FPGA scenario.

4. **Timing control:** The attacker can schedule query injections with coarse timing control (e.g., at fixed intervals) but does *not* have access to internal trigger signals from the HDC accelerator. Fine-grained alignment is obtained implicitly through staircase profiling (described in subsection 6.4.4).

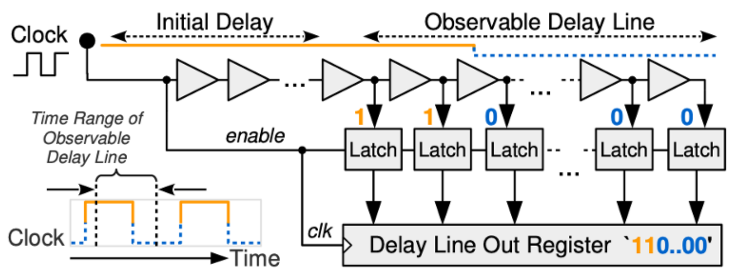


Figure 6.10.: TDC circuit used as an on-chip sensor [67]. A delay line, latches, and capture registers convert fine-grained voltage-induced delay variations into a digital value observable by the attacker.

We explicitly exclude any direct access to configuration bitstreams, training datasets, or internal memories. All information is derived solely from (i) publicly observable classifier outputs and (ii) timing variations measured by the on-chip TDC.

6.4.2. On-Chip Sensing with Time-to-Digital Converters

A TDC is a circuit that converts time (or delay) into a digital code and is extremely sensitive to supply voltage variations. In our context, it serves as an internal voltmeter that monitors the PDN shared with the HDC accelerator.

The core of the TDC is a delay line composed of a chain of buffers with latches (or flip-flops) tapped along the path. At every measurement:

- A clock edge is launched into the delay line.
- Latches are enabled at the start of the clock cycle and disabled halfway through the period.
- The number of latches that have toggled when sampling is performed encodes how far the clock edge has propagated, i.e., the effective delay.

Since the propagation delay of the buffers depends on the core voltage V_{int}, any voltage droop on the PDN, for example, when the HDC accelerator performs intensive XOR and POPCOUNT operations, slows down the signal, reducing the number of toggled latches. The captured thermometer-like code is then read out through registers and converted to a scalar TDC value.

A dedicated clock generator in the FPGA fabric produces different clock frequencies for the TDC and the HDC accelerator. The TDC typically runs at a higher frequency to increase temporal resolution, while the HDC uses a lower frequency optimized for inference latency and throughput. Both domains share the same PDN, so power-hungry HDC operations induce measurable shifts in the TDC readouts. The TDC architecture is illustrated in Figure 6.10.

6.4.3. Collision Analysis in High-Dimensional HDC

Collision analysis is a classical technique in side-channel cryptanalysis [11]. In traditional settings (e.g., AES), the attacker leverages known plaintext–ciphertext pairs and searches for distinct keys or intermediate values that produce identical outputs, thereby narrowing the key space and localizing leakage.

In HDC, we adapt this idea to the high-dimensional hypervector space. Because ClassHVs are constructed as distributed and redundant representations, distinct QueryHVs can produce highly similar similarity scores (i.e., small Hamming distance to a given ClassHV). We refer to such cases as *partial collisions*. These partial collisions are data-dependent and carry information about the underlying ClassHV.

Our key intuition is:

- QueryHVs that frequently yield the same predicted class and exhibit very low Hamming distance to that class's hypervector must be structurally close to the true ClassHV in hypervector space.

Algorithm 5 Collision-Based Hypervector Candidate Reduction

```
1: procedure CollisionReduce(HDC, CHV, $HV_{\text{dim}}$, percentile)
2:     $RandomHV \leftarrow$ GenerateRandomHVs($HV_{\text{dim}}$, numHV)
3:     $collisionCandidates \leftarrow [\,]$, $distances \leftarrow [\,]$
4:     for $i \leftarrow 1$ to numHV do
5:         $guessClass \leftarrow$ Classify(HDC, $RandomHV[i]$)
6:         if $guessClass$ = targetClass then
7:             $d \leftarrow$ HD($RandomHV[i]$, CHV)
8:             Append($collisionCandidates, RandomHV[i]$)
9:             Append($distances, d$)
10:    $\tau \leftarrow$ Percentile($distances$, percentile)
11:    $finalCandidates \leftarrow \{c \in collisionCandidates \mid \text{HD}(c, \text{CHV}) \leq \tau\}$
12:    return $finalCandidates$
```

- By identifying and analyzing these collision-prone QueryHVs, we can form a reduced candidate set that tightly constrains the bits of the secret ClassHV, dramatically reducing the search space for the subsequent TDC-based attack.

6.4.3.1. Collision-Based Hypervector Candidate Reduction

Given the high dimensionality (we use $D = 5000$), exhaustive exploration of the hypervector space is infeasible. To address this, we introduce a software-side collision-based candidate reduction that operates before any hardware side-channel measurement:

1. **Random candidate generation:** Generate a large pool (e.g., 100,000) of random hypervectors $HV_i \in \{0,1\}^D$ as QueryHVs.
2. **Classification and filtering:** For each candidate HV_i, obtain the predicted class label from the HDC accelerator. Retain only those candidates that are classified as the target class.
3. **Distance scoring:** For each retained candidate, compute its Hamming distance to a reference (e.g., an approximate or initial ClassHV) and record the distance.
4. **Percentile-based selection:** Choose the subset of candidates whose distance lies within the lowest $p\%$ (e.g., $p = 2\%$). These candidates form the collision-sensitive set that most closely resembles the true ClassHV.

5 summarizes the procedure.

In practice, this procedure reduces the candidate set from tens of thousands of random hypervectors to $O(10^3)$ high-quality candidates that are statistically very close to the true ClassHV. As shown later in subsection 6.4.7, this reduction significantly decreases the number of traces needed in the subsequent TDC-based attack.

6.4.4. Implicit Triggering via Staircase Profiling

A major challenge in remote internal attacks is synchronization: without an explicit trigger signal from the victim circuit, how can the attacker precisely align TDC measurements with the victim's HDC operations?

We address this using an implicit triggering technique based on *staircase profiling*. During a preliminary profiling phase, the attacker:

- injects carefully chosen QueryHVs at fixed intervals (e.g., every 100 ns, corresponding to a 10 MHz injection rate), and
- continuously records TDC outputs at a high sampling rate.

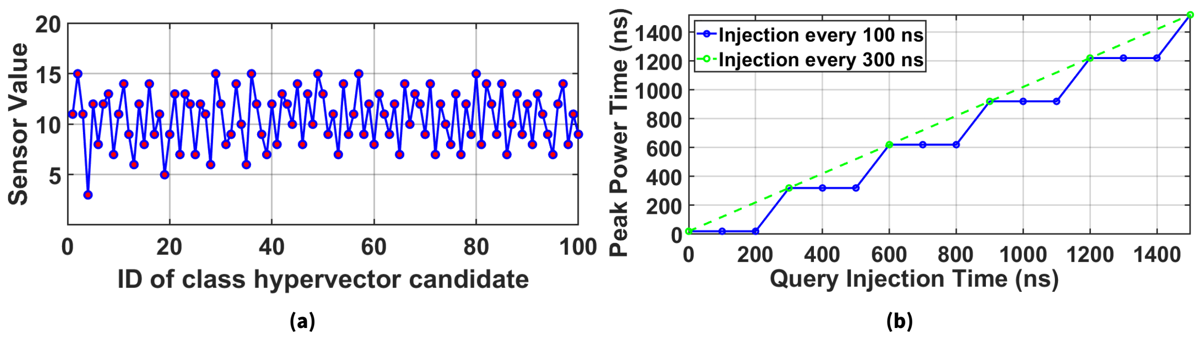


Figure 6.11.: (a) TDC values captured for different class candidates (MNIST) during profiling, demonstrating distinct peaks per candidate. (b) Query injection time vs. peak power timing with queries injected every 100 ns. The staircase pattern reveals the victim HDC execution period (300 ns) and the internal latency (≈ 20 ns offset) to reach peak power.

Because the injection pattern is asynchronous to the HDC accelerator's internal execution period, the phase difference between query injection and the start of HDC computation drifts over time. When plotting the time of observed TDC peaks against the injection index, a characteristic staircase pattern emerges:

- TDC peaks associated with HDC similarity computation appear periodically within each HDC execution window (e.g., every 300 ns).
- The absolute peak timing within the window (e.g., at 20 ns offset from the start) reveals the internal latency until the most power-hungry operations occur (such as XOR + POPCOUNT).

Figure 6.11 illustrates this behavior. The staircase pattern exposes both the HDC execution period (300 ns in our prototype) and the internal offset at which maximum power occurs. After this profiling, the attacker knows exactly when to sample the TDC within each execution window, effectively constructing a *virtual trigger* without any wired connection to the victim.

6.4.5. Two-Stage Internal SCA Methodology

Combining the collision-based reduction and implicit triggering, our internal SCA attack proceeds in two stages, summarized in Figure 6.12.

Stage 1: Collision-based candidate reduction (software-side). The attacker uses the collision analysis in subsubsection 6.4.3.1 to reduce the hypervector search space. This step:

- runs entirely off-chip using standard classifier outputs,
- yields a compact set of candidate QueryHVs that are strongly correlated with the target ClassHV, and
- reduces the number of hypervectors for which expensive TDC measurements must be collected.

Stage 2: TDC-based side-channel discrimination (hardware-side). Using the implicit trigger discovered via staircase profiling:

- The attacker injects reduced candidate QueryHVs into the HDC accelerator with precise timing.
- The TDC, clocked at one or more frequencies, records delay-line states at the time of peak HDC activity.
- Because XOR-based similarity computations consume less power when the candidate QueryHV is more similar (lower Hamming distance) to the true ClassHV, the TDC readings encode similarity information.

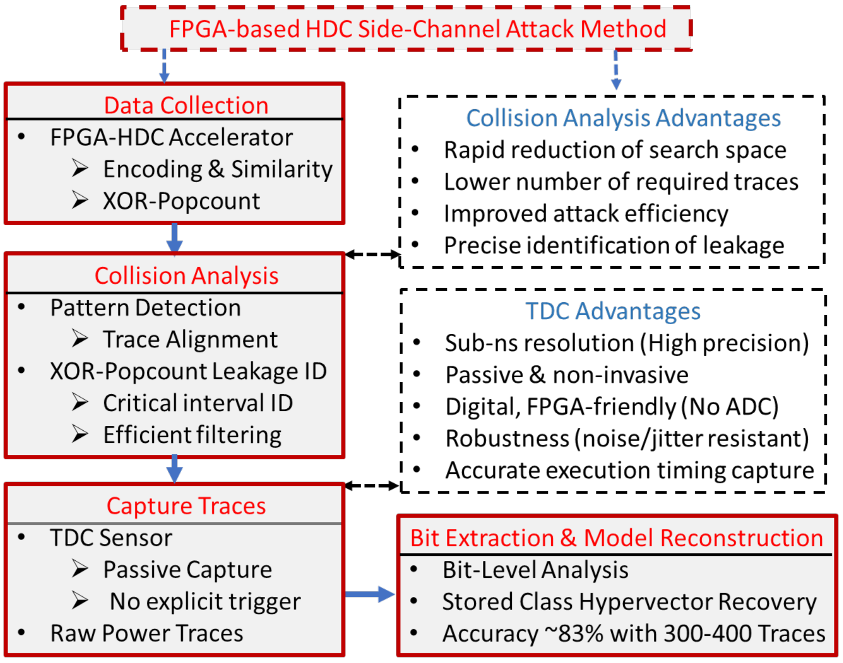
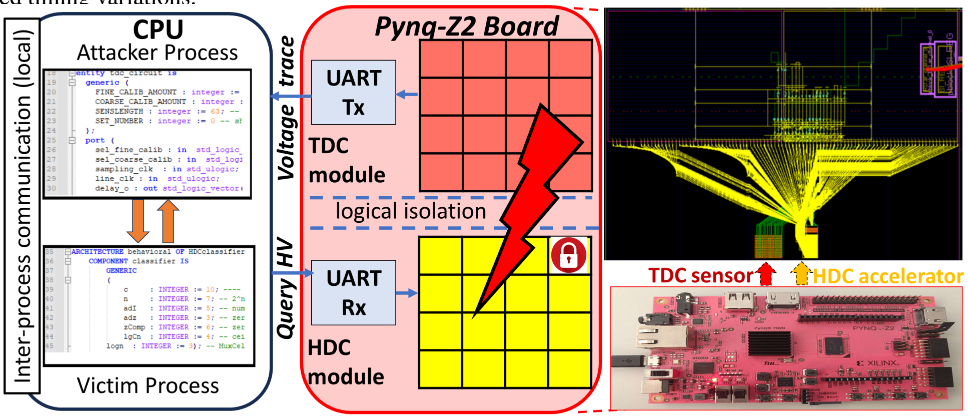

FPGA-based HDC Side-Channel Attack Method
Data Collection
• FPGA-HDC Accelerator
➢ Encoding & Similarity
➢ XOR-Popcount
Collision Analysis Advantages
• Rapid reduction of search space
• Lower number of required traces
• Improved attack efficiency
• Precise identification of leakage
Collision Analysis
• Pattern Detection
➢ Trace Alignment
• XOR-Popcount Leakage ID
➢ Critical interval ID
➢ Efficient filtering
TDC Advantages
• Sub-ns resolution (High precision)
• Passive & non-invasive
• Digital, FPGA-friendly (No ADC)
• Robustness (noise/jitter resistant)
• Accurate execution timing capture
Capture Traces
• TDC Sensor
➢ Passive Capture
➢ No explicit trigger
• Raw Power Traces
Bit Extraction & Model Reconstruction
• Bit-Level Analysis
• Stored Class Hypervector Recovery
• Accuracy ~83% with 300-400 Traces

Figure 6.12.: Proposed internal SCA methodology on an HDC accelerator. Stage 1 performs collision-based sensitivity analysis to reduce the hypervector search space. Stage 2 uses a TDC sensor and implicit triggering to distinguish candidate ClassHVs via power-induced timing variations.



```
entity tdc_circuit is
  generic (
    FINE_CALIB_AMOUNT : integer :=
    COARSE_CALIB_AMOUNT : integer :
    SENSLENGTH : integer := 63; --
    SET_NUMBER : integer := 0 -- sh
  );
  port (
    sel_fine_calib : in  std_logic_
    sel_coarse_calib : in  std_logi
    sampling_clk  : in std_ulogic;
    line_clk : in  std_ulogic;
    delay_o : out std_logic_vector(
```

```
ARCHITECTURE behavioral OF HDCclassifier
    COMPONENT classifier IS
        GENERIC
        (
            c      : INTEGER := 10; ----
            n      : INTEGER := 7; -- 2^n
            adI    : INTEGER := 5; -- num
            adz    : INTEGER := 3; -- zer
            zComp  : INTEGER := 6; -- zer
            lgCn   : INTEGER := 4; -- cei
        logn  : INTEGER := 3); -- MuxCel
```

Figure 6.13.: Floorplan of the proposed attack on Pynq-Z2. The HDC accelerator (victim, red) and TDC sensor (attacker, yellow) are placed in different clock regions within the same FPGA fabric but share the same PDN. Screenshot from Xilinx Vivado.

By comparing TDC outputs across candidates, the attacker infers which candidate bits are consistent with the true ClassHV. Aggregating these observations yields a bit-wise estimate of the secret ClassHV. As shown later, we achieve ≈ 83% bit recovery for a 5000-D ClassHV using this method.

6.4.6. Experimental Setup

We implement and evaluate the internal sensing attack on a Pynq-Z2 board. The Pynq-Z2 integrates a Xilinx Zynq XC7Z020 SoC combining an ARM-based processing system (PS) and programmable logic (PL) on the same die, which naturally supports both victim and attacker logic in one device.

Key aspects of the setup (illustrated in Figure 6.13):

- The HDC accelerator is implemented in one region of the PL and clocked at 24 MHz. It receives QueryHVs from a host PC via UART or AXI, performs similarity computation against stored ClassHVs, and returns the predicted class label.

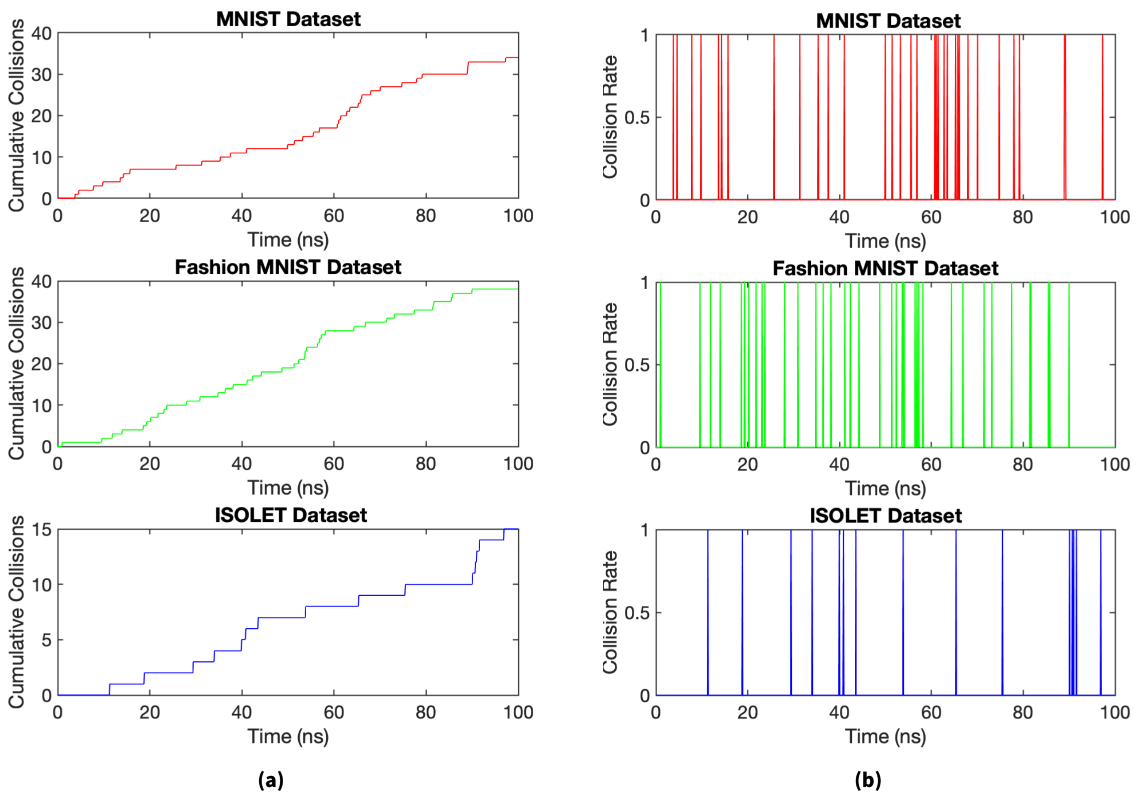


Figure 6.14.: (a) Cumulative number of collisions over time, showing an increasing trend as more query hypervectors are classified. (b) Collision frequency over time, with peaks indicating hypervectors that are particularly likely to align with the target ClassHV.

- The TDC sensor is implemented in a separate clock region, clocked at several frequencies (24, 48, 72, and 96 MHz) to probe different timing sensitivities. It shares the PDN with the HDC accelerator and outputs TDC codes through a UART interface.
- During profiling experiments, a debug trigger can optionally be asserted when the HDC module starts processing a QueryHV (for validation). However, the main attack uses only implicit timing derived from staircase profiling, making explicit triggers unnecessary in principle.

This arrangement reflects a multi-tenant scenario: the TDC core behaves like a small, hostile *co-located* IP that passively monitors PDN activity while the HDC accelerator performs normal inference.

6.4.7. Vulnerability Assessment of Unprotected HDC Designs

The TDC delay values directly reflect instantaneous power consumption: lower power leads to smaller voltage droop, shorter propagation delays, and thus a different TDC code. For similarity computations in HDC, we exploit the fact that:

- lower Hamming distance between QueryHV and ClassHV $\Rightarrow$ fewer bit flips in XOR logic $\Rightarrow$ lower dynamic power, and
- this manifests as a distinguishable pattern in the TDC readings.

Collision statistics. We first analyze collisions produced by Algorithm 5. Figure 6.14 shows:

- The cumulative collisions (Figure 6.14a) grow over time as more QueryHVs are evaluated, with distinct surges when hypervector patterns align strongly with the target ClassHV.
- The collision rate (Figure 6.14b) exhibits sharp peaks during phases of strong alignment. These peaks identify particularly informative QueryHVs that are later used as candidates in the TDC-based attack.

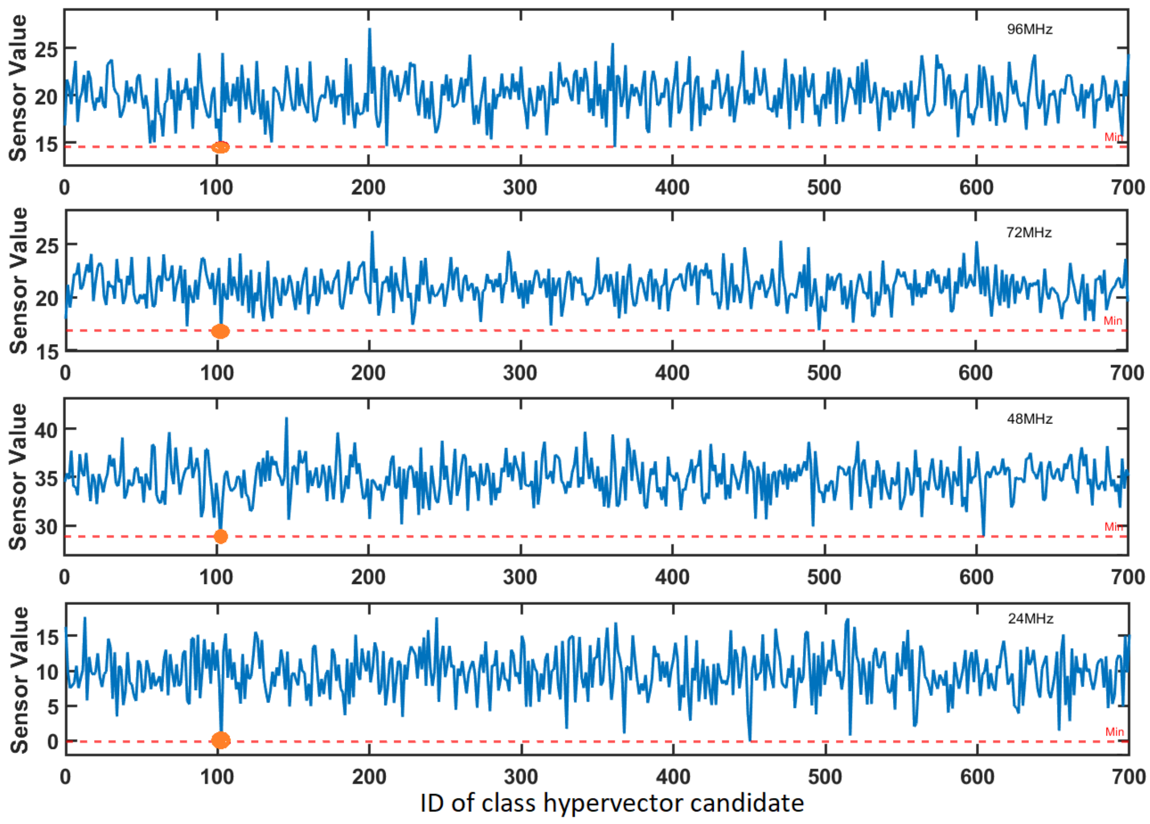


Figure 6.15.: TDC delay-line calibration for candidate ClassHVs on MNIST. The minimum TDC value (red circle) corresponds to the correct candidate (ID 102). Measurements are performed with TDC clocked at 24, 48, 72, and 96 MHz while the HDC remains at 24 MHz.

Table 6.5.: Recovered bits and inference accuracy for unprotected HDC designs.

Dataset	Inference acc. (before attack)	Bits recovered (% of ClassHV)	Inference acc. (after attack)
MNIST	81%	83.3%	78%
F-MNIST	79%	79.8%	73%
ISOLET	84%	73.0%	78%

Overall, we extract on the order of 1000 candidate ClassHV IDs as promising targets before performing any side-channel measurement.

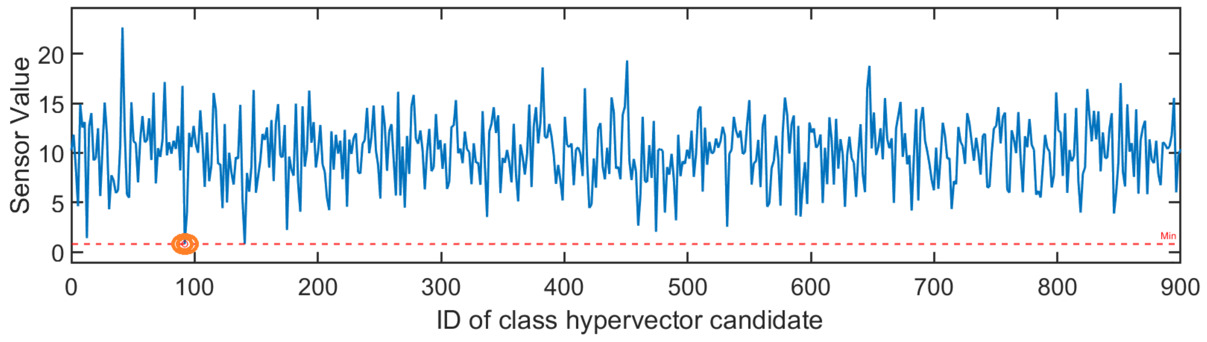
TDC calibration and bit recovery. We then apply the TDC-based attack (Stage 2) on the reduced candidate set. Figure 6.15 illustrates the TDC delay calibration across several candidate IDs for the MNIST dataset.

The correct candidate hypervector consistently produces the lowest TDC code across multiple TDC frequencies, enabling reliable identification. Using this approach, we recover a 5000-D ClassHV for MNIST with 83.3% of bits correct.

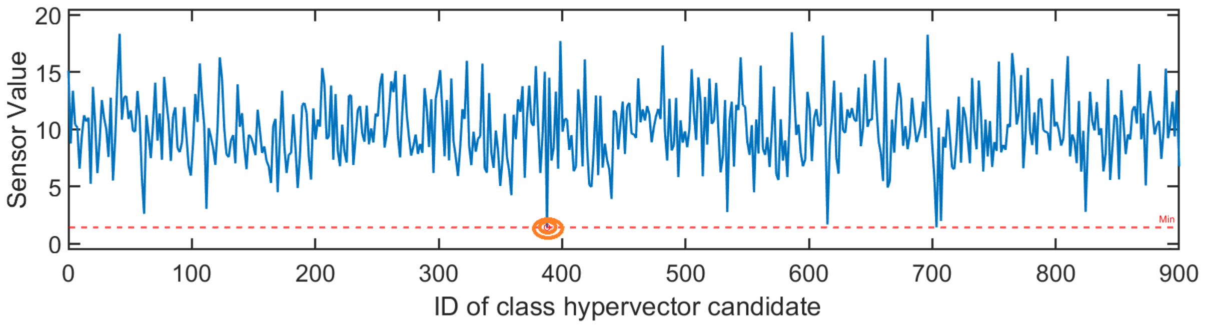
We further validate the attack on Fashion-MNIST (F-MNIST) and ISOLET, as shown in Figure 6.16.

Impact on inference accuracy. To quantify the practical impact of bit recovery, Table 6.5 reports the percentage of recovered bits and corresponding inference accuracy before and after the ClassHV is reconstructed using the attack.

Even though the recovered ClassHVs contain only 73–83% correct bits, the resulting inference accuracy remains close to the baseline (within 3–6 percentage points). This confirms that partial hypervector recovery is sufficient for practical model theft.



(a) Fashion-MNIST: minimum TDC value (red marker) at candidate ID 96.



(b) ISOLET: minimum TDC value (red marker) at candidate ID 388.

Figure 6.16.: Successful TDC-based power attacks on recovered ClassHVs for Fashion-MNIST and ISOLET when both HDC and TDC modules operate at 24 MHz.

6.4.8. Randomization-Based Countermeasure

Ensuring security in resource-constrained edge AI systems is challenging. Conventional countermeasures like heavy-weight masking, large dummy loads, or constant-power logic are often impractical on FPGAs due to area, power, and latency overheads. We therefore propose a lightweight randomization strategy tailored to HDC accelerators.

The core idea is to decorrelate the HDC's instantaneous switching activity from the true data-dependent similarity computation by injecting controlled randomness into encoding and memory access patterns. Concretely:

- A small LFSR-based pseudo-random number generator is instantiated on-chip.
- Its output is used to generate pseudo-random masks that are XORed with intermediate hypervectors and/or memory addresses during both training and inference.
- The masks are chosen so that the final decision logic (e.g., after unmasking or compensating operations) preserves the correct classification result, but the instantaneous power consumption becomes much less correlated with the true QueryHV–ClassHV similarity.

As shown in Figure 6.17, randomization:

- preserves classification accuracy (baseline and protected curves overlap), and
- drastically reduces the maximum bit recovery accuracy to about 19%, effectively neutralizing the TDC-based attack.

The additional hardware cost of the LFSR and a small number of XOR gates is modest, making this countermeasure attractive for edge deployments.

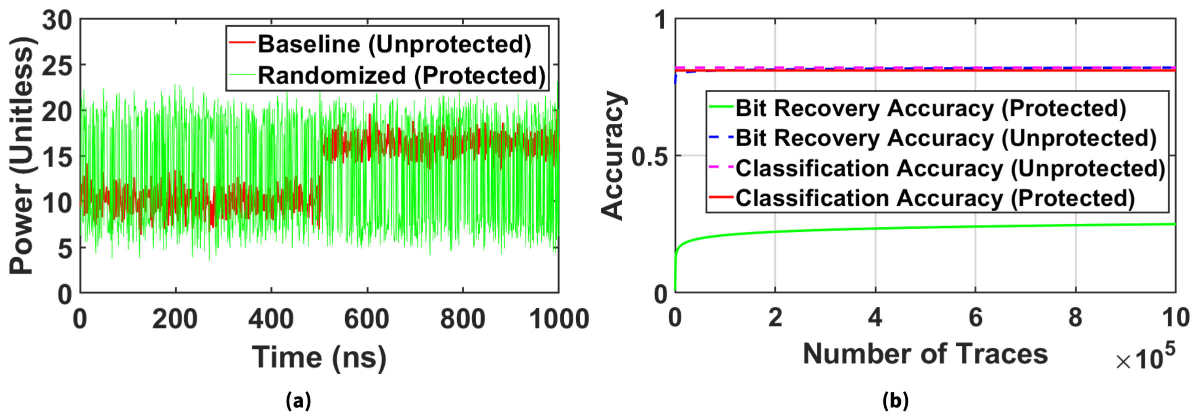


Figure 6.17.: Effect of randomization on MNIST. (a) Sample power patterns of HDC operations with and without randomization: the baseline (blue) shows predictable patterns, while the randomized design (green) significantly disrupts correlation and peak structure. (b) Classification accuracy (black, overlapping for protected and unprotected designs) and maximum bit recovery accuracy for a single class: randomization reduces recovery from ≈ 80–83% (blue) to ≈ 19% (green).

Table 6.6.: Comparative analysis of the proposed TDC-based internal SCA with representative attacks on HDC.

Aspect	**Yang et al. [81]**	**Hernandez et al. [84]**	**Liu et al. [94]**	**Krautter et al. [93]**	**This work (TDC-based SCA)**
Attack type	Adversarial input	Model inversion	Adversarial parameter attack	Voltage/timing stress FI	Internal side-channel attack
Target	HDC classifier	FPGA-HDC	FPGA-HDC	FPGA-HDC	FPGA-HDC
Attack method	Genetic algorithm	Feature reconstruction	Bit flipping (RowHammer-like)	Voltage/thermal stress	TDC timing and power analysis
Vulnerability exploited	Inference input space	Associative memory semantics	Associative memory encoding	Implementation timing margins	Similarity operation leakage (XOR+POPCOUNT)
Key outcome (%)	Misclassification (∼ 78)	Input reconstruction (∼ 81)	Misclassification (∼ 90)	Accuracy degradation	Bit recovery (∼ 83)
Trigger mechanism	Adversarial inputs	Reconstruction queries	Parameter perturbation	Voltage/thermal variation	Profiled timing triggering (implicit)
Countermeasures evaluated	Adversarial training	Noise injection, quantization	–	–	Randomization-based hiding

6.4.9. Comparison with Other Attacks on HDC

Finally, Table 6.6 compares our internal TDC-based SCA with other prominent attacks on HDC classifiers. While adversarial-input and parameter-based attacks focus on inducing misclassification or corrupting associative memories, our approach directly targets hardware-level leakage of the similarity computation.

In summary, the internal TDC-based attack complements the external DL-SCA together, they show that both external power measurements and internal timing sensors can be exploited to steal or clone HDC models. This underscores the need for side-channel-aware design and lightweight, HDC-specific countermeasures for secure edge AI deployment.

6.5. Active Integrity Attacks: Targeted Voltage-Level Fault Injection (HyFault)

The third attack vector, *HyFault*, shifts the focus from confidentiality (IP theft) to *integrity* (targeted misclassification). While HDC is often cited for its robustness against random errors (e.g., bit flips due to soft errors), we show that it is highly vulnerable to *targeted* voltage-level fault injection under precise timing information modeling. By combining boundary-sensitive profiling with cycle-accurate power analysis and optimized glitching, an attacker can force specific misclassifications with success rates approaching 90%, even in high-dimensional HDC implementations.

6.5.1. Threat Model

Attack Capability

We consider an attacker with non-invasive physical access to the target HDC accelerator, for example in a lab setting or on a physically exposed edge device. The attacker:

- can control the supply voltage of the FPGA using a fault injection platform such as ChipWhisperer Pro (CW1200), enabling precise voltage glitches at nanosecond time scales;

- has detailed knowledge of the HDC architecture and inference pipeline (encoding, similarity computation, memory organization) but *no* direct access to internal associative memory storing sensitive ClassHV bits (proprietary IP of the HDC accelerator);
- is constrained by realistic limitations: restricted measurement time, limited physical access sessions, and noisy measurement conditions.

Attack Goals

The attacker's primary goal is to induce *targeted* misclassification by injecting voltage-level faults during critical phases of HDC inference. The injected faults cause controlled bit flips or timing violations that perturb similarity scores, thereby forcing specific wrong labels (e.g., digit "4" → "9"). Unlike traditional cryptographic fault attacks that recover keys, our threat model targets the functional integrity of HDC models (e.g., safety-critical classification outcomes) [60], [77], [85].

6.5.2. Profiling and Sensitivity Analysis

HDC's redundancy and distributed representation confer robustness to *random* faults: flipping a few random bits seldom changes the predicted class [100], [107]. As a result, naively glitching at arbitrary times or on arbitrary queries rarely leads to misclassification. To mount an effective attack, we must (i) identify input hypervectors that lie close to decision boundaries and (ii) locate timing windows where small perturbations have maximal impact. We consider hypervector dimensionalities of 1000-D and 2000-D, which are representative for practical edge deployments.

6.5.2.1. FPGA-Based Boundary Detection

To avoid exhaustive power profiling on the victim device (which is resource-intensive), we use a reference FPGA from the same family as the target (e.g., CW305) for rapid boundary screening. Each input image I is encoded into a query hypervector Q_I. The HDC accelerator computes similarity scores between Q_I and each stored class hypervector C_i using Hamming similarity:

$$S(Q_I, C_i) = D - \text{HammingDistance}(Q_I, C_i), \tag{6.1}$$

where D is the hypervector dimensionality.

We define a confidence margin for input I as the difference between the top-two similarity scores:

$$R(I) = S_{\text{best}}(Q_I, C_i) - S_{\text{second-best}}(Q_I, C_i). \tag{6.2}$$

Inputs with low $R(I)$ lie close to the decision boundary between two classes. We declare an input *boundary-sensitive* if

$$R(I) \leq \tau, \tag{6.3}$$

where the threshold $\tau = \mu - 1.5\sigma$ is computed empirically from the distribution of confidence scores (mean μ, standard deviation σ). We then minimally perturb the corresponding hypervectors Q_I (e.g., flipping $\approx 0.5\%$ of bits) and collect power traces while the FPGA recomputes similarity. Correlation power analysis (CPA) on these traces identifies hypervectors and time windows with high sensitivity to small perturbations. The full procedure is summarized in Algorithm 6. Figure 6.18 shows a typical confidence distribution for 100k randomly generated inputs in 1000-D and 2000-D configurations; the threshold τ (pink) selects boundary-sensitive images. We then perform CPA to locate sensitive hypervectors and time windows. As shown in Figure 6.19, a correlation threshold of 0.15 identifies roughly 500 highly sensitive hypervectors, while the power trace reveals the similarity computation window as the most vulnerable region due to its strong data-dependent activity.

Algorithm 6 Boundary Detection for Sensitive Hypervectors

```
 1: Input: Image set $\{I\}$
 2: Output: Boundary-sensitive image set $\mathcal{B}$
 3: $\mathcal{B} \leftarrow \emptyset$
 4: for each image $I$ in $\{I\}$ do
 5:     Encode $I \longrightarrow Q_I$
 6:     Compute similarities $S(Q_I, C_i)$ for all classes $i$
 7:     Find $S_{\text{best}}$ and $S_{\text{second-best}}$
 8:     $R(I) \leftarrow S_{\text{best}} - S_{\text{second-best}}$
 9: Compute $\mu, \sigma$ from $\{R(I)\}$ and set $\tau = \mu - 1.5\sigma$
10: for each $I$ with $R(I) \leq \tau$ do
11:     Minimally perturb $Q_I$ ($\approx 0.5\%$ bits)
12:     Collect power traces; perform CPA to confirm sensitivity
13:     $\mathcal{B} \leftarrow \mathcal{B} \cup \{I\}$
14: return $\mathcal{B}$
```

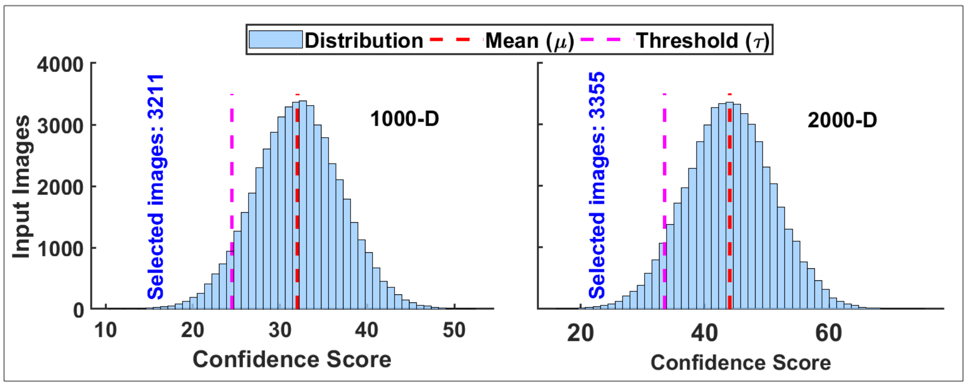


Figure 6.18.: Confidence score distribution for 100k random input hypervectors (1000-D and 2000-D). The adaptive threshold τ (pink) selects boundary-sensitive inputs near the decision boundary.

6.5.2.2. Sensitivity vs. Dimensionality

Boundary detection yields a comparable *number* of sensitive input patterns across 1000-D and 2000-D settings, but the *effort* required for successful fault injection grows with dimensionality. HDC's robustness is enhanced in higher dimensions: more bits encode the same semantic information, and more bit flips are needed to cross the decision boundary.

In our experiments:

- For 1000-D hypervectors, perturbations of 2–5 bits ($\approx 0.5\%$) per sensitive hypervector are sufficient to cause misclassification.
- For 2000-D hypervectors, perturbations of 5–10 bits (still $\approx 0.5\%$) are required to achieve similar misclassification behavior.

This dimensional scaling provides a useful baseline when interpreting the success rates of HyFault across different HDC configurations.

6.5.3. Fault Injection Attack Methodology

Given the set of sensitive inputs and vulnerable timing windows, we now describe the voltage-level fault injection methodology. The attack proceeds in two main stages: Random fault injections within vulnerable windows to map coarse sensitivity and optimization of glitch timing (other parameters too) to maximize misclassification.

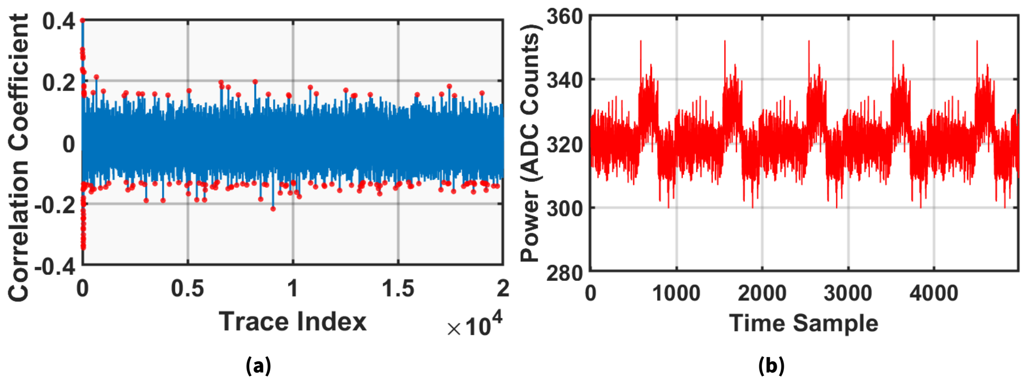


Figure 6.19.: Profiling and sensitivity analysis on 1000-D HDC (MNIST). (a) CPA correlation coefficients; hypervectors above the 0.15 threshold (red line) are highly sensitive and suitable for targeted FI. (b) Representative power trace showing three phases: encoding, similarity computation, and writing. The similarity phase exhibits the highest data-dependent variance and becomes the primary target for fault injection.

6.5.3.1. Random Fault Injection within Vulnerable Windows

From the power traces in Figure 6.19(b), we identify the similarity computation region as the most fault-sensitive phase. We initially inject random glitches in this window, using standardized parameters from prior work [65], [87], such as:

- glitch width $\approx$ 20 ns,
- near-zero offset from the start of the similarity phase.

We sweep candidate timestamps across this phase and perform 10–40 injections per point. For each timestamp, we record:

- the fraction of successful faults (any effect),
- the fraction causing correct classification,
- the fraction causing wrong classification (our desired outcome),
- the fraction causing resets or crashes.

This coarse sweep reveals regions where glitches are more likely to cause misclassification, which become the basis for fine-grained timing optimization.

6.5.3.2. Optimization of Fault Injection Timing and Parameters

To achieve *targeted* misclassification with high probability, we model the fault injection process and derive an optimal timestamp t^*. For a set of candidate timestamps $T = \{t_1, t_2, \ldots, t_n\}$, we empirically estimate:

$$P_{\text{fault}}(t_i) = \frac{\text{Successful faults at } t_i}{\text{Total trials at } t_i}, \tag{6.4}$$

and conditional probabilities of classification outcomes given a fault:

$$P_{\text{CC}|\text{fault}}(t_i) + P_{\text{WC}|\text{fault}}(t_i) + P_{\text{NC}|\text{fault}}(t_i) = 1, \tag{6.5}$$

where CC, WC, and NC denote correct, wrong, and no classification, respectively.

The overall misclassification probability at t_i is:

$$P_{\text{WC}}(t_i) = P_{\text{fault}}(t_i) \cdot P_{\text{WC}|\text{fault}}(t_i), \tag{6.6}$$

and we also consider the *transition* probability from correct to wrong classification:

$$P_{\text{WC,trans}}(t_i) = P_{\text{WC}|\text{fault}}(t_i) \cdot P_{\text{CC}}(t_i). \tag{6.7}$$

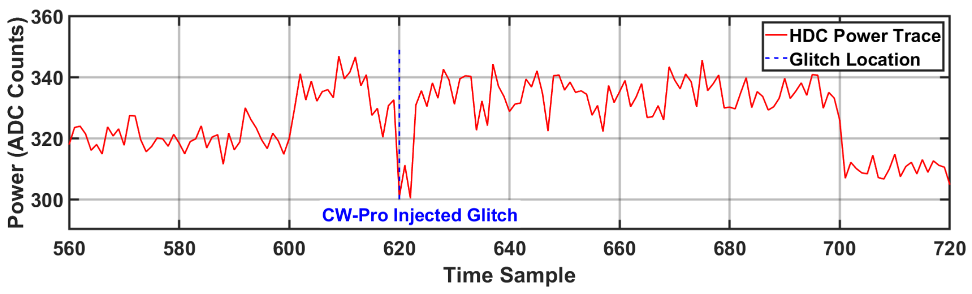


Figure 6.20.: ChipWhisperer Pro trace with a single voltage glitch (blue) at 622 ns within the similarity computation window (MNIST). This window is identified as highly fault-sensitive via profiling.

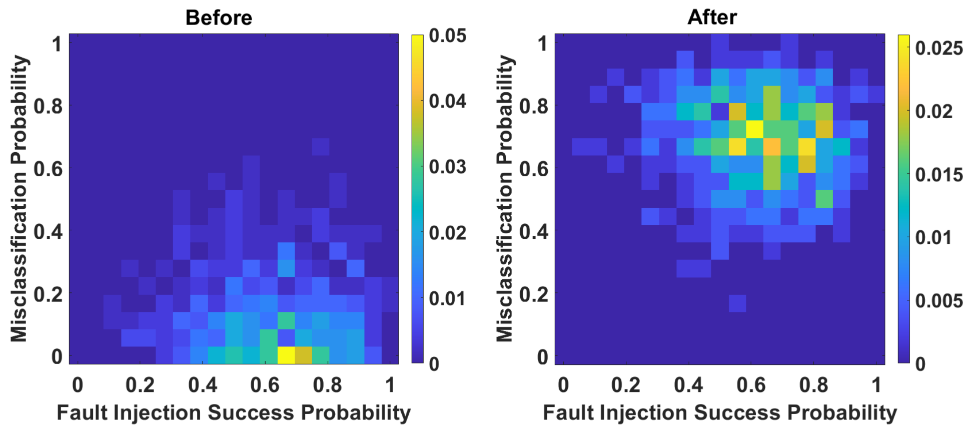


Figure 6.21.: Comparison of fault injection effectiveness before and after parameter optimization (1000-D MNIST). After tuning glitch width, offset, and repetition count, successful faults more consistently result in misclassification rather than resets or benign behavior.

We choose the optimal timestamp t^* by maximizing:

$$t^* = \arg\max_{t_i} \left(P_{\text{fault}}(t_i) \cdot P_{\text{WC|fault}}(t_i) + P_{\text{WC,trans}}(t_i) \right). \tag{6.8}$$

In practice, we first identify candidate windows via the coarse sweep, then refine around these windows with 5 ns steps and optimized glitch widths and offsets.

6.5.3.3. Empirical Verification with ChipWhisperer Pro

We empirically validate the model using the ChipWhisperer Pro (CW1200) platform. Figure 6.20 shows a power trace captured from the CW305 Artix-7 FPGA under a representative 1000-D MNIST workload, with a glitch injected at 622 ns during the similarity phase.

Initially, using nominal glitch parameters (20 ns width, zero offset), we observe modest misclassification rates. After optimizing glitch timing and parameters, misclassification probabilities increase sharply, as shown in Figure 6.21 and Table 6.7.

Glitch widths of 25–30 ns with offsets around -15 to -20 ns and higher repetition counts give misclassification rates up to 89% while keeping reset rates relatively low.

6.5.4. Countermeasures for Voltage Fault Injection

To mitigate HyFault, we design countermeasures that disrupt the attacker's ability to correlate timing and power signatures with specific similarity operations, without incurring prohibitive overheads.

6.5.4.1. Hypervector Randomization

Hypervector randomization hides regularity in power and timing by permuting hypervector dimensions at runtime. We implement:

Table 6.7.: Effect of Glitch Parameters on Misclassification (MNIST)

Voltage Glitch Parameters			**Injection Outcome**	
Offset (ns)	**Width (ns)**	**Repeat**	**Misclassified (%)**	**Reset (%)**
-10	20	10	0	9
-15	20	10	4	15
-20	20	10	9	22
-10	25	20	23	36
-15	25	20	61	31
-20	25	20	74	23
-10	30	40	71	14
-15	30	40	**89**	11
-20	30	40	**89**	2
-10	35	40	85	14
-15	35	40	83	13
-20	35	40	83	16

- a lightweight LFSR-based PRNG to generate pseudo-random permutation indices;
- precomputed permutation tables for ClassHVs, applied once at initialization;
- dynamic permutations for QueryHVs during inference.

These permutations preserve functional correctness (decoding is unaffected), but destroy the repeatability of per-dimension switching patterns. As a result, the timing map between bit positions and glitch timestamps becomes unstable, making targeted glitches far less effective.

6.5.4.2. Dual *XOR* Masking

To further strengthen the similarity computation against faults, we introduce dual *XOR* masking. Masked class hypervectors are computed offline as:

$$H_{\text{class\_masked}} = H_{\text{class}} \oplus M_{\text{class}}, \tag{6.9}$$

while query hypervectors are dynamically masked:

$$H_{\text{query\_masked}} = H_{\text{query}} \oplus M_{\text{query}}. \tag{6.10}$$

Similarity is then computed on masked values:

$$H_{\text{result}} = H_{\text{query\_masked}} \oplus H_{\text{class\_masked}}. \tag{6.11}$$

For additional robustness, we use two independent masks per query and class:

$$H_{\text{masked1}} = (H_{\text{query}} \oplus M_{\text{query1}}) \oplus (H_{\text{class}} \oplus M_{\text{class1}}), \tag{6.12}$$
$$H_{\text{masked2}} = (H_{\text{query}} \oplus M_{\text{query2}}) \oplus (H_{\text{class}} \oplus M_{\text{class2}}), \tag{6.13}$$

with masks refreshed by a PRNG. This splits the effective computation into two shares that must remain consistent; a fault in one path tends to decorrelate the shares, yielding noise rather than a clean misclassification.

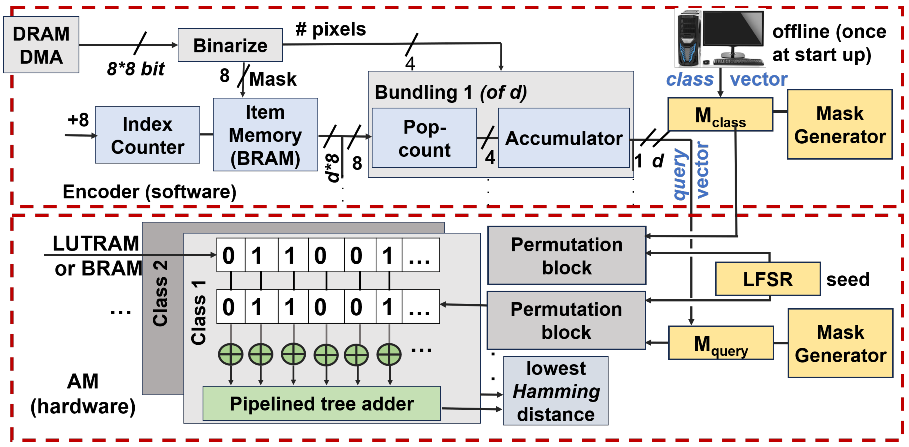


Figure 6.22.: Countermeasure structure combining LFSR-based permutation randomization and dual XOR masking. Class hypervectors are masked and permuted offline; query hypervectors are dynamically masked and permuted at runtime, complicating precise glitch timing and profiling.

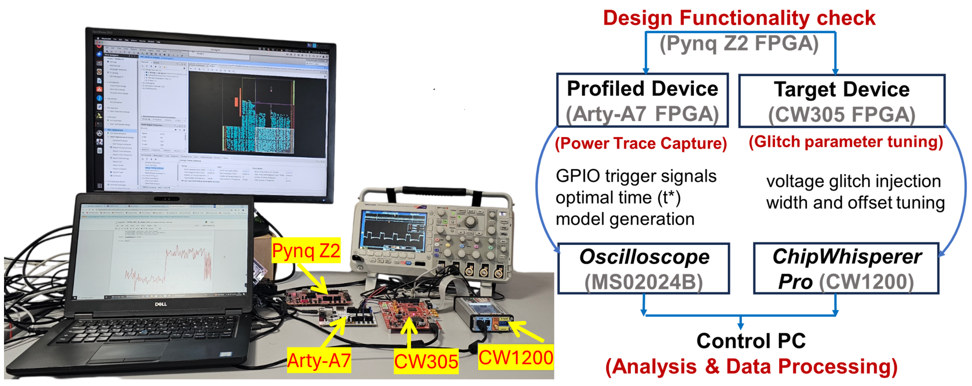


Figure 6.23.: Experimental setup: control PC, Pynq Z2 and Arty-A7 FPGA boards, ChipWhisperer Pro (CW1200) for voltage glitching, and CW305 Artix-7 FPGA target board for precise characterization of HyFault.

6.5.5. Evaluation and Results

6.5.5.1. Experimental Setup

Figure 6.23 shows the integrated experimental framework. A baseline, unprotected BRAM-based HDC accelerator is implemented on a Pynq Z2 FPGA (100 MHz), synthesized using Xilinx Vivado 2023.1. Classification outputs are sent via UART to a host PC for validation. For fault injection, we use the CW1200 with a CW305 FPGA board. Due to API constraints, we port the validated HDC design from an Arty-A7 to CW305 via a compatible bitstream. Profiling on Arty-A7 identifies the similarity computation cycles (e.g., cycles 62–76) as most sensitive; glitches on CW305 are then synchronized to these cycles. Glitch parameters (offset, width, amplitude) are iteratively tuned under realistic attacker constraints.

6.5.5.2. Profiling and Sensitivity (Quantitative)

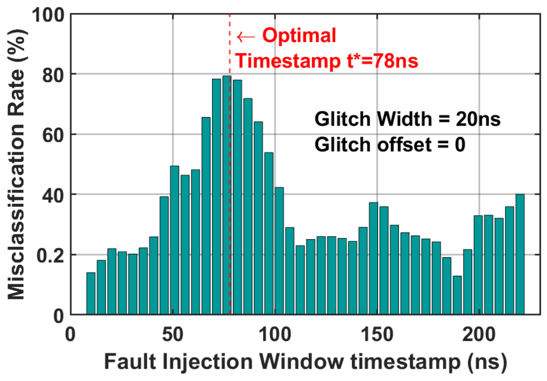
Using the ChipWhisperer Pro, we perturb 3,300 sensitive inputs, flipping 2–5 bits per 1000-D hypervector, and collect $\approx$ 20,000 target power traces. CPA confirms strong correlation between specific hypervectors and trace windows (see Figure 6.19), and a correlation threshold of 0.15 selects the top 500 hypervectors as highly sensitive. This reduces the subsequent glitching effort substantially.

6.5.5.3. Targeted Misclassification Across Datasets

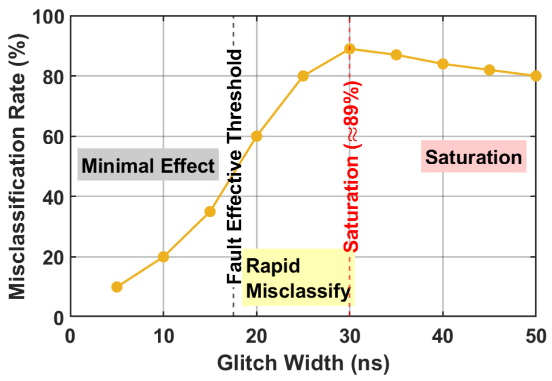
By exploiting intrinsic similarity between certain classes (e.g., visually similar digits or neighboring letters), HyFault can steer predictions from an original class to a chosen target class. Table 6.8 summarizes

Table 6.8.: Targeted misclassification results across benchmark datasets using HyFault.

Dataset	Original	Target	Sim. Diff. (%)	Misclass. (%)
MNIST	4	9	1.5	85
	3	8	2.1	80
	1	7	2.3	78
ISOLET	B	D	2.3	78
	M	N	2.7	76
	F	S	3.0	72
Cardio	Normal	Arrhythmia	1.8	82
	Arrhythmia	Normal	2.0	79
CIFAR-10	Cat	Dog	2.1	75
	Automobile	Truck	2.4	72
HAR	Walking	Upstairs	1.6	80
	Upstairs	Downstairs	1.9	77



(a) Misclassification rate as a function of fault-injection timestamp. A clear peak emerges at the optimal injection point near $t^\star \approx 78$ ns, indicating maximum sensitivity of the similarity-computation phase.



(b) Effect of glitch width on misclassification behavior. Small widths produce minimal impact, rapid growth appears between 20–30 ns, and misclassification rate saturates near 89% beyond 30 ns.

Figure 6.24.: Empirical analysis for optimal fault injection. (a) Misclassification probability across candidate timestamps (5 ns increments); the optimal region near 78 ns is highlighted. (b) Misclassification vs. glitch width, showing minimal effect at small widths, rapid growth between 20–30 ns, and saturation beyond 30 ns.

results across benchmark datasets. Class pairs with small similarity-score differences ($\approx$ 1.5–3%) are especially vulnerable: precisely timed glitches during similarity computation can reliably redirect predictions.

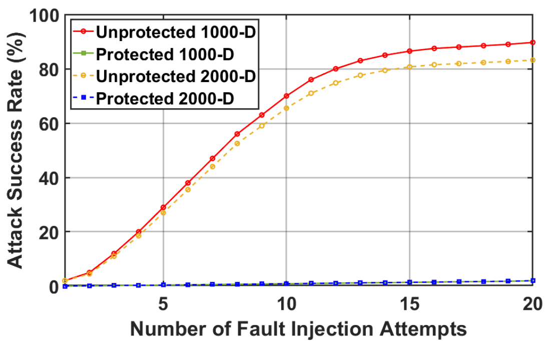
6.5.5.4. Timing and Glitch Parameter Optimization

We empirically map misclassification probability vs. glitch timestamp and width. Figure 6.24a shows the probability of misclassification as a function of injection time (5 ns step). The most vulnerable timestamp is around 78 ns relative to the start of the similarity window.

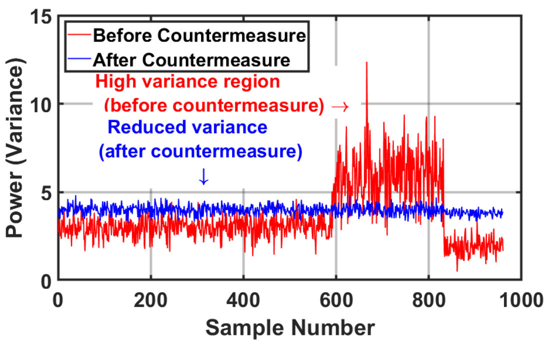
Combining timing and width sweeps yields the optimized parameters in Table 6.7, achieving misclassification rates up to 89% on 1000-D MNIST with acceptable reset rates.

6.5.5.5. Impact of Hypervector Dimensionality

We evaluate HyFault on 1000-D and 2000-D HDC implementations. Increasing dimensionality reduces fault injection success: misclassification rates decrease by $\approx$ 3 percentage points for MNIST when moving from 1000-D to 2000-D, due to greater redundancy and dispersion of information. This effect is further reinforced when countermeasures are applied (see below).

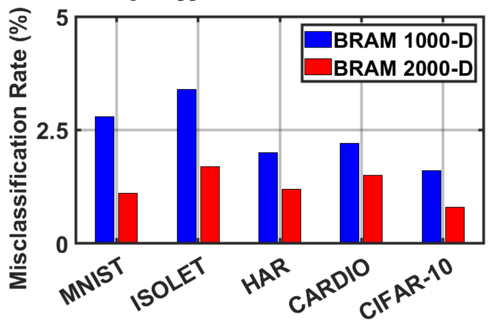


(a) Attack success rate vs. number of fault-injection attempts for 1000-D and 2000-D HDC designs. Countermeasures reduce misclassification rates from nearly 90% to below 3%, demonstrating strong resistance to targeted fault injection.

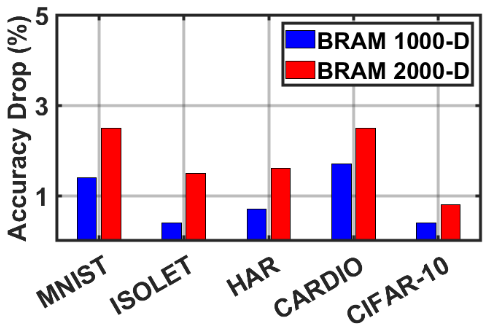


(b) Power-trace variance comparison. The unprotected design (red) shows high variance caused by data-dependent switching activity, while the protected design (blue) exhibits significantly lower and more uniform variance, confirming reduced leakage.

Figure 6.25.: Comparison of unprotected and protected HDC designs on MNIST under voltage-level fault injection. (a) Attack success rate as a function of the number of glitch attempts for 1000-D and 2000-D hypervectors, showing that countermeasures suppress misclassification from ~89% down to ≤2–3%. (b) Power-trace variance before and after protection: the unprotected design (red) exhibits strong data-dependent fluctuations, whereas the protected design (blue) drastically reduces variance, indicating effective leakage suppression.



(a) Misclassification rates across datasets for 1000-D and 2000-D BRAM-based HDC. Increasing dimensionality (2000-D) consistently lowers misclassification probability, demonstrating improved robustness under protection.



(b) Classification accuracy drop after applying countermeasures. Accuracy degrades only slightly (2–3%), confirming that protection achieves strong security with minimal impact on model performance.

Figure 6.26.: Effect of hypervector dimensionality and countermeasures on BRAM-based HDC accelerators. (a) Higher dimensionality (2000-D) further suppresses misclassification rates across all datasets, especially in protected designs. (b) The resulting accuracy loss remains small (2–3%), demonstrating that the proposed countermeasures provide strong robustness with minimal performance overhead.

6.5.5.6. Effectiveness of Proposed Countermeasures

We apply the combined hypervector randomization and dual XOR masking to the BRAM-based HDC design and repeat targeted fault injection experiments. Figure 6.25 compares success rates and power variance before and after protection. Across datasets, misclassification rates fall to ≤ 2–3% under protection, effectively returning the system to near-native reliability.

The effect of dimensionality under protection is shown in Figure 6.26: higher dimensionality further suppresses attack success, especially in BRAM-dominated implementations.

Table 6.9 summarizes area, timing, and power overheads of the countermeasures.

Overheads are modest: BRAM increases by 12.7–16.3%, FFs by 4.9–5.8%; latency increases 11.2–14.5%, throughput decreases 6.4–8.9%, and power grows only 2.8–3.1%. Classification accuracy drops insignificantly (≈ 2–3%), making the scheme practical for resource-constrained edge deployments.

6.5.5.7. Number of Power Traces Required

Typical cryptographic SCA often requires hundreds of thousands to millions of traces to extract fine-grained leakage. In contrast, the HDC classifier exhibits strong, localized leakage during similarity computation. Our profiling and sensitivity analysis require only ≈ 20,000 traces to identify sensitive hypervectors

Table 6.9.: Area, timing, and power overhead of countermeasures (Artix-7 BRAM-based HDC).

Dim.	**Resource Utilization ↑ (%)**		**Timing Overhead (%)**		**Power ↑ (%)**
	BRAM	**FF**	**Latency ↑**	**Throughput ↓**	
1000-D	12.7	4.9	11.2	6.4	2.8
2000-D	16.3	5.8	14.5	8.9	3.1

Table 6.10.: Comparative analysis of HyFault with representative HDC attacks.

Aspect	**This Work (HyFault)**	**Yang et al. [81]**	**Hernandez et al. [84]**	**Liu et al. [94]**	**Krautter et al. [93]**
Attack Type	Fault injection	Adversarial input	Model inversion	Adversarial parameter	Voltage/timing stress
Target	FPGA-HDC	HDC simulation	FPGA-HDC	FPGA-HDC	FPGA-HDC
Method	Profiling + voltage glitch	Genetic algorithm	Feature reconstruction	Bit flipping (RowHammer-like)	Voltage/thermal/timing stress
Vulnerability Phase	Similarity computation	Inference input space	Model encoding	Associative memory	Implementation logic
Success Rate	**89.7%** misclassification	78% misclassification	High reconstruction accuracy	~89% misclassification	Significant accuracy drop
Countermeasures	Masking + randomization	Adversarial training	Noise injection, quantization	–	–

and optimal glitch timings. This relatively small trace count underscores the practical vulnerability of FPGA-based HDC to fault injection. Table 6.10 compares HyFault with representative HDC attacks. HyFault is unique in combining boundary profiling, precise glitch timing, and hardware-level evaluation to achieve near-deterministic misclassification on FPGA-HDC.

6.5.6. Integrated Countermeasures and the Robustness

6.5.6.1. Dynamic Masking Against DL-SCA

To address the deep-learning-based power SCA from section 6.3, we employ *lightweight dynamic masking*. Hypervectors are XORed with pseudo-random masks M generated by an on-chip PRNG:

- Stored ClassHVs are masked: $C' = C \oplus M$.
- Query QueryHVs are masked with the same or related mask: $Q' = Q \oplus M$.
- Similarity is computed on (C', Q'); linearity of XOR allows the effective mask to cancel or be compensated, preserving correctness while randomizing instantaneous Hamming weights.

This reduces ResNet-34-based bit extraction accuracy from 93% to $\approx$ 18%, as validated by higher-order Test Vector Leakage Assessment (TVLA): first-order t-values drop from 9.7 (severe leakage) to 2.2 (below 4.5 threshold), indicating that both first- and second-order leakage are effectively suppressed. Overhead is modest: $\approx 1.6\times$ LUTs and $\approx 1.4\times$ latency, considerably lighter than traditional cryptographic masking.

6.5.6.2. Hypervector Randomization and Dual Masking Against FI/Collision

For collision-based remote SCA (section 6.4) and HyFault, hypervector randomization and dual XOR masking (see Figure 6.22) jointly obfuscate timing and power signatures. Query permutation breaks the attack's implicit timing map, while dual masking converts precise faults into largely random noise. Combined, these defenses reduce misclassification success from $\sim$ 89.7% to $\sim$ 2.1%, effectively restoring HDC's natural robustness. The summary of the trade-offs in our proposed countermeasures is addressed in Table 6.11.

6.5.6.3. The Robustness Paradox in HDC

The distributed and redundant nature of hypervectors allows HDC classifiers to tolerate substantial levels of *random* noise: flipping even hundreds of bits rarely alters the predicted class because information is encoded holistically across thousands of dimensions. This characteristic is often cited as evidence of HDC's robustness. However, our results reveal that this robustness is highly asymmetric and holds only for *stochastic*, unstructured perturbations. When an adversary introduces *targeted* faults at precisely chosen cycles, locations, or logic paths, this redundancy becomes a liability. Small but strategically aligned perturbations, for example, flipping 5–10 specific bit positions in the XOR or POPCOUNT

Table 6.11.: Qualitative summary of countermeasures: target, mechanism, overhead, and efficacy.

Countermeasure	Target Attack	Mechanism	Area / Time	Efficacy
Dynamic masking	DL-SCA (CNN)	XOR with PRNG mask	~1.6× LUTs, 1.4× latency	93% → 18% bit acc.
Query randomization	FI & collision SCA	Dimension permutation	Minimal / negligible	89.7% → 2.1% succ.
Dual XOR masking	FI & collision SCA	Redundant computation	Moderate / low	89.7% → 2.1% succ.

datapath—can deterministically bias similarity scores, collapse class-decision margins, and force consistent misclassifications without triggering resets or detectable anomalies. We refer to this phenomenon as the robustness paradox: the same distributed representation that masks random noise also masks the attacker's controlled perturbations, enabling stealthy integrity violations.

Crucially, all three attack vectors studied in this thesis converge on the *similarity computation unit* (XOR + POPCOUNT) as the dominant architectural hotspot. In DL-SCA, the massive parallel switching of the XOR array produces high-SNR, data-dependent power signatures that a deep CNN can learn to exploit. In dynamic workload scheduling and TDC-based SCA, the high instantaneous current draw of POPCOUNT operations induces measurable timing shifts on the shared PDN, enabling on-chip sensors to infer Hamming-distance variations remotely. In HyFault, the tight timing margins of the POPCOUNT adder tree make it highly susceptible to precisely timed voltage glitches, which selectively corrupt similarity accumulation while leaving the remaining HDC pipeline unaffected. Taken together, these observations show that while HDC is resilient to random noise at the algorithmic level, the hardware realization of similarity computation exposes a concentrated and repeatable vulnerability—a clear focal point for future hardening and secure microarchitectural design.

6.5.7. Summary

This chapter presented a comprehensive security evaluation of FPGA-based HDC accelerators, spanning three hardware-level attack vectors:

- **Deep-learning-assisted power SCA (DL-SCA):** A 1D ResNet-34, combined with adaptive Grad-CAM, learns non-linear leakage features from power traces, achieving up to 93% bit recovery on ClassHVs across datasets and FPGA platforms.
- **Ddynamic workload scheduling/internal sensing SCA (collision-based TDC attacks):** By combining collision analysis with on-chip TDC sensors, an attacker can recover ≈ 83% of ClassHV bits without physical probing, exploiting PDN timing fluctuations.
- **Voltage-level fault injection (HyFault):** Using boundary profiling, timing optimization, and ChipWhisperer-based glitching, targeted misclassification rates up to 89.7% are achieved on BRAM-based HDC accelerators, with realistic glitch parameters.

Collectively, these results establish a baseline of insecurity: unprotected FPGA-HDC accelerators are vulnerable to both IP theft and integrity compromise. At the same time, we show that these vulnerabilities are not fundamental. Lightweight, HDC-aware countermeasures, such as dynamic masking, hypervector randomization, and dual XOR masking, can reduce attack success rates to below 3% with modest overhead in area, latency, and power, and minimal impact on accuracy.

As edge AI scales and HDC becomes more widely deployed, algorithmic robustness alone is insufficient as a security guarantee. The transition from "robust by design" to *secure by design* requires explicit hardware hardening, side-channel-aware architectures, and standardized countermeasures. Future work should aim at formal verification of side-channel resilience for HDC accelerators, integration of these defenses into tool flows, and extending protections to emerging edge platforms, ensuring that the energy efficiency and robustness of hyperdimensional computing do not come at the cost of security.

7. Conclusion and Outlook

Summary of the Thesis

This dissertation developed a cross-layer, *experimentally validated* framework for the physical security of emerging AI hardware accelerators. The work began by formalizing physics-aware leakage and fault models for analog CiM architecture, then demonstrated real-hardware attacks and defenses on digital/FPGA accelerators, HDC, and finally generalized the methodology to neuromorphic substrates. The contributions close long-standing gaps between device physics, circuit-level non-idealities, and system-level security evaluation, and they translate into concrete, low-overhead countermeasures suitable for edge deployment.

Contributions.

- **Analog CiM (ReRAM/MRAM) leakage modeling (Ch. 3):** The thesis provides a device-to-array-to-PDN model that explains how analog non-idealities, captured by the $\mathbf{v}^{\top}\Delta\mathrm{G}(t)\mathbf{v}$ term, propagate to supply/EM observables. The model yields windowed predictors for compute primitives (scouting-logic XOR; MAC+tanh; CAM/HDC) and predicts traces-to-disclosure (N_{TD}) trends under PDN decoupling and technology corners, enabling *design-time* leakage sign-off.
- **Persistent faults in MRAM-backed digital systems (Ch. 4):** On real hardware, the thesis shows that rail-isolated, nanosecond-scale voltage glitches during the STT-MRAM commit window induce persistent, byte-local corruptions in stored keys. The resulting constancy enables low-data-complexity cryptanalysis (DFA/SPFA): a full AES-128 key is recovered with only 12–17 correct/faulty pairs, two orders of magnitude below typical volatile-memory DFA requirements. PUF-sealed key slots, randomized commit timing, and write-verify/ECC neutralize this class with modest overhead.
- **Emerging Neuromorphic computing (Ch. 5):** For a-IGZO-based flexible SNNs, the work introduces *FlexSpy*, a design-time framework that identifies a dominant leakage primitive: *rate-weighted quasi-DC offsets* during spike epochs. Using only a single shunt trace and a self-generated "virtual trigger," the attacker achieves label inference up to ROC–AUC = 0.91 and recovers layer-wise firing rates with NMSE = 0.14. Two lightweight defenses, spike-time jitter and event balancing, reduce measurable leakage by 38–70% at $\leq 9\%$ power cost.
- **FPGA HDC: real-hardware SCA & FI (Ch. 6):** The work demonstrates three complementary attacks on HDC accelerators. (i) A deep-learning SCA (1D ResNet-34 with adaptive Grad-CAM) attains up to 93% bit recovery of ClassHVs; (ii) a remote, TDC-based timing/power attack ("Collide & Conquer") reaches 83% recovery *without* external probes; and (iii) a targeted voltage-glitch attack (HyFault) forces up to 89.7% misclassification. Defenses, dynamic masking, and hypervector randomization with dual XOR masking cut recovery/misclassification down to ~18% and ~2%, respectively, with ~1.6× LUT and ~1.4× latency overheads for SCA hardening and single-digit percent overheads for FI hardening.

Synthesis: From Physics to Systems to Defenses

A unifying physical thread

Across CiM arrays, flexible neuromorphics, and FPGA HDC, one mechanism dominates: *data-dependent current draw filtered by a shared power-delivery network*. Whether the signal is continuous (analog CiM) or impulsive (CMOS switching in HDC), the PDN converts internal activity into measurable supply/timing signatures. Conversely, precisely timed supply perturbations convert PDN susceptibility into *fault* channels (droop-induced delay or mixed-signal threshold shifts). This duality explains why the same unit (XOR+popcount similarity in HDC; sense/activation in CiM; synaptic epochs in f-SNNs) becomes the focal point for both SCA and FI.

Summary of Methods

The thesis establishes a repeatable methodology: (i) leakage/fault *priors* grounded in device physics; (ii) windowed predictors tied to compute primitives; (iii) rigorous statistics (TVLA, $|\rho|$, N_{TD}, MI, SLI); and (iv) controlled instrumentation stacks on boards and in simulation. This progression, from formal predictor to measurement to mitigation, turns "security by afterthought" into *security by construction*.

What the results imply for edge AI

Three system-level insights emerge.

1. HDC's tolerance to random faults masks fragility to *targeted* faults and aligned leakages; flexible SNN sparsity, prized for energy efficiency, becomes a strong quasi-DC side channel. Robustness must be *recast* as *adversarial robustness* at the hardware boundary.
2. Internal monitors (e.g., TDCs) and shared resources that exist for reliability and DVFS become high-fidelity side-channel sensors under multi-tenancy; their use must be threat-modeled and access-controlled.
3. Per-inference masking/permutation, start-time jitter, and randomized commit timing are *cheap* yet highly effective, because they *break alignment* (SCA) and *destroy determinism* (FI/PFA) while preserving functional equivalence.

Answers to the Research Questions (RQs)

RQ1 (Device → Leakage/Fault link). The derived models and simulations show how ReRAM/MRAM conductances and WL/BL parasitics set windowed predictors that correlate with measured traces; commit-window asymmetries in STT-MRAM produce *persistent* faults under nanosecond-scale droops.
RQ2 (Beyond classical SCA for FPGA-HDC). Deep models discover temporally localized leakage (XOR/popcount) that classical CPA cannot, achieving up to **93%** bit recovery and transferring across FPGA boards; dynamic masking reduces TVLA from ~9.7 to ~2.2 and bit recovery to ~18%.
RQ3 (FI operating regime). Voltage-glitch FI succeeds when aligned to similarity windows and BRAM read phases, achieving **up to 89.7%** targeted misclassification; dual-share/permute defenses push success down to ~2% while keeping throughput and power impacts small.
RQ4 (PDN covert activation & joint hardening). On-chip timing sensors (TDC) and controlled query cadence enable *implicit triggering* and remote SCA with **83%** recovery; the same PDN-centricity motivates *joint* defenses (masking, permutation, start-time jitter, monitoring) that simultaneously blunt SCA and FI.

Design Guidance

1. Treat XOR+popcount (HDC), sense/activation (CiM), and synaptic epochs (SNN) as *security-critical IP*. Apply per-inference masking/permutation and bounded start-time jitter; avoid storing or transporting unmasked hypervectors.
2. Use *time hopping* (few-cycle, seed-driven jitter), randomized scheduling of CAM/ML reads, and, for NVM, PUF-keyed *commit dithering*.
3. Lightweight rail monitors (RO/XADC) should gate writes and freeze outputs under droop; track TVLA/MI/SLI as regression checks alongside timing and power sign-off.
4. Seal persistent assets (keys, weights, hypervectors) to a device-unique PUF with integrity tags; add dual-slot/ECC with write-verify and short-TTL caches for sensitive artifacts.
5. Scope and rate-limit access to on-chip sensors (e.g., TDC/ILA) under multi-tenancy; isolate critical rails and avoid public triggers correlated with sensitive commit windows.

Limitations

- **Adversary model.** Most empirical results assume laboratory- or gray-box capabilities (board access or co-tenancy). While widely relevant to edge devices and cloud FPGAs, fully remote adversaries were not the focus.
- **Scope of channels.** The strongest results center on power/timing channels; EM and thermal channels, while discussed, warrant deeper empirical coverage for flexible substrates and dense CiM arrays.
- **Defense composition.** While the thesis quantifies single/paired countermeasures with resource budgets, formal *composability guarantees* (e.g., proofs under adaptive attackers) remain open.

Outlook and Future Work

1. Integrate TVLA/MI/SLI targets into Neural Architecture Search (NAS) and HDC encoder design; co-train models that *maximize* functional accuracy while *minimizing* hardware leakage predictability.
2. Elevate security metrics to first-class sign-off alongside STA and power closure (e.g., “security timing corners” with PDN droop models and leakage budgets).
3. Provide lightweight, verifiable entropy sources for per-inference masking/jitter on low-end FPGAs and flexible substrates.
4. Develop access-control and attestation for on-chip monitors (TDC/XADC/thermal) to prevent their repurposing as covert sensors under multi-tenancy.
5. Extend FlexSpy-like methodologies to EM channels and to other emerging technology (PCM, FeFET CiM; 3D-integrated fabrics), including aging-aware leakage evolution.

This thesis establishes that the physical layer is the *root* of both vulnerability and defense in edge AI hardware. By turning device physics into quantitative predictors, measurements into reproducible benchmarks, and randomness into structured protection, the work charts a practical path to *hardware-rooted trust*. The proposed techniques, from per-inference masking/jitter to countermeasure, reduce successful attacks from the 80–90% range to single digits with acceptable overheads, and they generalize across analog CiM, flexible neuromorphics, and FPGA HDC. As edge intelligence becomes ubiquitous, the methodolgies developed here, models, metrics, and mitigations, provide a foundation for designing accelerators whose efficiency does not come at the expense of security.

Part III.

Appendix

Algorithm 7 Test Vector Leakage Assessment (TVLA) Using Welch's t-Test

```
Require: Two sets of leakage traces: $\mathcal{L}_0 \in \mathbb{R}^{N_0 \times T}$ (fixed / class 0), $\mathcal{L}_1 \in \mathbb{R}^{N_1 \times T}$ (random / class 1);
 1: transform $g(\cdot)$ (identity for first order, centered square for second order);
 2: significance threshold $\tau$ (typically $\tau = 4.5$ for first order).
Ensure: $t$-statistic vector $\mathbf{t} \in \mathbb{R}^T$ and leakage mask $\boldsymbol{\ell} \in \{0,1\}^T$.
                              ▷ Optional: apply non-linear transform for higher-order TVLA
 3: for $i = 1$ to $N_0$ do
 4:     for $t = 1$ to $T$ do
 5:         $Y_{i,t}^{(0)} \leftarrow g(\mathcal{L}_0[i,t])$
 6: for $i = 1$ to $N_1$ do
 7:     for $t = 1$ to $T$ do
 8:         $Y_{i,t}^{(1)} \leftarrow g(\mathcal{L}_1[i,t])$
                              ▷ Compute Welch's $t$-statistic per time sample
 9: for $t = 1$ to $T$ do
10:     $\mu_0[t] \leftarrow \frac{1}{N_0} \sum_{i=1}^{N_0} Y_{i,t}^{(0)}$
11:     $\mu_1[t] \leftarrow \frac{1}{N_1} \sum_{i=1}^{N_1} Y_{i,t}^{(1)}$
12:     $\sigma_0^2[t] \leftarrow \frac{1}{N_0-1} \sum_{i=1}^{N_0} (Y_{i,t}^{(0)} - \mu_0[t])^2$
13:     $\sigma_1^2[t] \leftarrow \frac{1}{N_1-1} \sum_{i=1}^{N_1} (Y_{i,t}^{(1)} - \mu_1[t])^2$
14:     $se[t] \leftarrow \sqrt{\sigma_0^2[t]/N_0 + \sigma_1^2[t]/N_1}$          ▷ Standard error
15:     if $se[t] > 0$ then
16:         $t[t] \leftarrow \dfrac{\mu_0[t] - \mu_1[t]}{se[t]}$
17:     else
18:         $t[t] \leftarrow 0$
19:     $\ell[t] \leftarrow \mathbf{1}(|t[t]| \geq \tau)$          ▷ Leakage flag
20: return $\mathbf{t}, \boldsymbol{\ell}$
```

Algorithm 8 Mutual Information and Spike-Leakage Index Estimation

```
Require: Discrete secrets $\{S_n\}_{n=1}^{N}$, $S_n \in \mathcal{S}$;
 1: Leakage feature vectors $\{\mathbf{z}_n\}_{n=1}^{N}$ for a given block $b$ and window $w$;
 2: number of histogram bins $B$ per feature dimension;
 3: (optional) device-level features $\{\mathbf{z}_n^{\text{dev}}\}_{n=1}^{N}$ for two-layer MI.
Ensure: $I(S;Z)$ in bits, $\text{SLI}(b,w;S) \in [0,1]$;
 4: (optional) two-layer decomposition $I(S;L_{\text{device}})$ and $I(S;L_{\text{design}} \mid L_{\text{device}})$.
                                              ▷ Step 1: discretize leakage features by binning
 5: Define bin edges for each dimension of $\mathbf{z}$ using min/max or quantiles.
 6: for $n = 1$ to $N$ do
 7:     $s \leftarrow S_n$
 8:     $q \leftarrow \textsc{Bin}(\mathbf{z}_n)$        ▷ Map $\mathbf{z}_n$ to a discrete bin index $q \in \{1,\dots,Q\}$
 9:     $C_S[s] \leftarrow C_S[s] + 1$
10:     $C_Z[q] \leftarrow C_Z[q] + 1$
11:     $C_{SZ}[s,q] \leftarrow C_{SZ}[s,q] + 1$
                                              ▷ Step 2: convert counts to probabilities
12: $N \leftarrow \sum_{s\in\mathcal{S}} C_S[s]$
13: for all $s \in \mathcal{S}$ do
14:     $p_S[s] \leftarrow C_S[s]/N$
15: for all $q$ with $C_Z[q] > 0$ do
16:     $p_Z[q] \leftarrow C_Z[q]/N$
17: for all $(s,q)$ with $C_{SZ}[s,q] > 0$ do
18:     $p_{SZ}[s,q] \leftarrow C_{SZ}[s,q]/N$
                                              ▷ Step 3: compute entropy of the secret
19: $H_S \leftarrow 0$
20: for all $s \in \mathcal{S}$ with $p_S[s] > 0$ do
21:     $H_S \leftarrow H_S - p_S[s]\log_2 p_S[s]$
                                              ▷ Step 4: compute mutual information $I(S;Z)$
22: $I \leftarrow 0$
23: for all $(s,q)$ with $p_{SZ}[s,q] > 0$ do
24:     $I \leftarrow I + p_{SZ}[s,q]\log_2\left(\frac{p_{SZ}[s,q]}{p_S[s]\,p_Z[q]}\right)$
                     ▷ Step 5: compute Spike-Leakage Index (SLI) for block $b$, window $w$
25: if $H_S > 0$ then
26:     $\text{SLI}(b,w;S) \leftarrow I/H_S$
27: else
28:     $\text{SLI}(b,w;S) \leftarrow 0$
       ▷ Optional: two-layer MI decomposition $I(S;L_{\text{device}})$ and $I(S;L_{\text{design}} \mid L_{\text{device}})$
29: if device-level features $\mathbf{z}_n^{\text{dev}}$ are provided then
30:     Estimate $I(S;L_{\text{device}})$ using the above steps with $(S_n, \mathbf{z}_n^{\text{dev}})$.
31:     Form concatenated features $\tilde{\mathbf{z}}_n \leftarrow [\mathbf{z}_n^{\text{dev}}, \mathbf{z}_n]$.
32:     Estimate $I(S;[L_{\text{device}}, L_{\text{design}}])$ using $(S_n, \tilde{\mathbf{z}}_n)$.
33:     $I(S;L_{\text{design}} \mid L_{\text{device}}) \leftarrow I(S;[L_{\text{device}}, L_{\text{design}}]) - I(S;L_{\text{device}})$
34: return $I$, $\text{SLI}(b,w;S)$ (and optional two-layer MI terms)
```

Algorithm 9 Correlation Power Analysis (CPA)

```
Require: Power traces $\{T_k(t)\}_{k=1}^N$, inputs $\{x_k\}_{k=1}^N$, hypothesis space $\mathcal{W}$ for secret $w$, window templates
    $\{g_w(t)\}$, leakage predictors $\{\text{PredictLeakage}_w(\cdot)\}$
Ensure: Estimated secret $\hat{w}$, per-window correlation scores $\rho_w(w_h)$
 1: for all hypotheses $w_h \in \mathcal{W}$ do
 2:     for $k = 1$ to $N$ do
 3:         for all windows $w$ do
 4:             Compute predicted leakage
                    $$p_k^{(w)} \leftarrow \text{PredictLeakage}_w(x_k, w_h)$$
 5:             Extract scalar feature from the trace via projection
                    $$y_k^{(w)} \leftarrow \int T_k(t)\, g_w(t)\, \mathrm{d}t$$
                                                        ▷ Matched filter / window integration
 6:     for all windows $w$ do
 7:         Compute Pearson correlation between predicted and measured leakage:
                $$\rho_w(w_h) \leftarrow \text{corr}(\{y_k^{(w)}\}_{k=1}^N,\ \{p_k^{(w)}\}_{k=1}^N)$$
 8:     Define hypothesis score
                $$S(w_h) \leftarrow \max_w |\rho_w(w_h)|$$
 9: Select best hypothesis
                $$\hat{w} \leftarrow \mathop{arg\,max}_{w_h \in W} S(w_h)$$
10: Compute traces-to-disclosure estimate using dominant correlation
                $$\rho_\star \leftarrow \max_{w_h \in \mathcal{W},\, w} |\rho_w(w_h)|, \qquad N_{\text{TD}} \approx \frac{z^2}{\rho_\star^2}$$
                                        ▷ $z$ is the desired confidence parameter (e.g., $z = 5$)
```

Bibliography

[1] R. E. Lyons and W. Vanderkulk, “The use of triple-modular redundancy to improve computer reliability”, *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[2] W. A. Wulf and S. A. McKee, “Hitting the memory wall: Implications of the obvious”, in *ACM SIGARCH Computer Architecture News*, vol. 23, ACM, 1995, pp. 20–24.

[3] R. J. Anderson and M. G. Kuhn, “Tamper resistance—a cautionary note”, *USENIX Workshop on Electronic Commerce*, 1996.

[4] D. Boneh, R. A. DeMillo, and R. J. Lipton, “On the importance of checking cryptographic protocols for faults”, in *EUROCRYPT*, ser. LNCS, vol. 1233, Springer, 1997, pp. 37–51.

[5] W. Maass, “Networks of spiking neurons: The third generation of neural network models”, *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[6] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis”, in *Advances in Cryptology—CRYPTO’99*, Springer, 1999, pp. 388–397.

[7] NIST, *Announcing the advanced encryption standard (aes)*, FIPS PUB 197, 2001.

[8] S. Chari, J. R. Rao, and P. Rohatgi, “Template attacks”, in *CHES 2002*, ser. LNCS, vol. 2523, Springer, 2002, pp. 13–28.

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

[10] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice Hall, 2003.

[11] K. Schramm, T. Wollinger, and C. Paar, “A new class of collision attacks and its application to des”, in *Fast Software Encryption: 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers 10*, Springer, 2003, pp. 206–222.

[12] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model”, in *CHES 2004*, ser. LNCS, vol. 3156, Springer, 2004, pp. 16–29.

[13] J. Mirković and P. Reiher, “A taxonomy of ddos attacks and ddos defense mechanisms”, *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[14] W. Schindler, K. Lemke, and C. Paar, “A stochastic model for differential side channel cryptanalysis”, in *CHES*, ser. LNCS, vol. 3659, Springer, 2005, pp. 30–46.

[15] S. Skorobogatov, “Semi-invasive attacks—a new approach to hardware security analysis”, Ph.D. dissertation, University of Cambridge, 2005.

[16] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, “The sorcerer’s apprentice guide to fault attacks”, in *Proceedings of the IEEE*, vol. 94, 2006, pp. 370–382.

[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley, 2006.

[18] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[19] F.-X. Standaert and C. Archambeau, “Using subspace-based template attacks to compare and combine power and electromagnetic information leakages”, in *CHES 2008*, ser. LNCS, vol. 5154, Springer, 2008, pp. 411–425.

[20] P. Kanerva, “Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors”, *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[21] P. Kanerva, “Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors”, *Cognitive computation*, vol. 1, no. 2, pp. 139–159, 2009.

[22] D. Ielmini, F. Nardi, and C. Cagli, “Resistance-dependent amplitude of random telegraph-signal noise in resistive switching memories”, *Applied Physics Letters*, vol. 96, no. 5, p. 053 503, 2010.

[23] F.-X. Standaert et al., “Information-theoretic and security evaluation of side-channel attacks and countermeasures”, *Proceedings of the IEEE*, vol. 103, no. 11, pp. 1–29, 2010.

[24] F.-X. Standaert, “Information-theoretic approaches to security evaluation of cryptographic devices under side-channel attacks”, *Proceedings of the IEEE*, vol. 98, no. 12, pp. 188–204, 2010.

[25] M. Doulcier-Verdier, J.-M. Dutertre, J. Fournier, J.-B. Rigaud, B. Robisson, and A. Tria, “A side-channel and fault-attack resistant aes circuit working on duplicated complemented values”, in *ISSCC*, IEEE, 2011.

[26] G. Goodwill et al., “A testing methodology for side-channel resistance validation”, in *NIST Non-Invasive Attack Testing Workshop*, 2011.

[27] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, “Machine learning in side-channel analysis: A first study”, *Journal of Cryptographic Engineering*, vol. 1, no. 4, pp. 293–302, 2011.

[28] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, and et al., “Neuromorphic silicon neuron circuits”, *Frontiers in Neuroscience*, vol. 5, p. 73, 2011.

[29] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. Springer, 2011.

[30] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Pearson, 2011.

[31] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines”, in *International Conference on Machine Learning (ICML)*, 2012.

[32] A. Dehbaoui, J.-M. Dutertre, K. Mirbaha, M. Renauld, A. Tria, and B. Robisson, “Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller”, in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2012, pp. 77–88.

[33] D. M. Harris and S. L. Harris, *Digital Design and Computer Architecture*, 2nd ed. Morgan Kaufmann, 2012, ISBN: 9780123944245.

[34] M. Joye and M. Tunstall, Eds., *Fault Analysis in Cryptography*. Springer, 2012.

[35] D. Apalkov, B. Dieny, and J. M. Slaughter, “Spin-transfer torque magnetic random access memory (stt-mram)”, *J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, p. 13, 2013.

[36] M. Horowitz, “1.1 computing’s energy problem (and what we can do about it)”, in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, IEEE, 2014, pp. 10–14.

[37] M. Horowitz, “1.1 computing’s energy problem and what we can do about it”, in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2014, pp. 10–14.

[38] S. Ambrogio and et al., “Statistical fluctuations in hfox resistive-switching memory: Part i—set/reset variability”, *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3812–3820, 2015.

[39] F. Bernard-Granger, B. Dieny, R. Fascio, and K. Jabeur, “SPITT: A magnetic tunnel junction SPICE compact model for STT-MRAM”, in *DATE*, 2015.

[40] Y. Chen, E. Keogh, et al., *The ucr time series classification archive*, `https://www.cs.ucr.edu/~eamonn/time_series_data/`, Accessed: 2025-11-18, 2015.

[41] A. D. Kent and D. C. Worledge, "Spin-transfer-torque mram: Challenges and prospects", *Nature Nanotechnology*, vol. 10, pp. 187–191, 2015.

[42] A. D. Kent and D. C. Worledge, "A new spin on MRAM", *Nature Nanotechnology*, vol. 10, no. 3, pp. 187–191, 2015.

[43] D. Apalkov, B. Dieny, and J. M. Slaughter, "Spin-transfer torque magnetic random access memory (STT-MRAM)", *Proceedings of the IEEE*, vol. 104, no. 10, pp. 1796–1830, 2016.

[44] P. Chi et al., "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory", *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 27–39, 2016.

[45] *ISO/IEC 17825:2016, testing methods for the mitigation of non-invasive attack classes against cryptographic modules*, International Organization for Standardization, Standard specification, 2016.

[46] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition", in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, IEEE, 2016, pp. 1–8.

[47] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing", in *Proceedings of the 2016 international symposium on low power electronics and design*, 2016, pp. 64–69.

[48] A. Shafiee et al., "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars", *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[49] F. Tramér, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis", in *USENIX Security Symposium*, 2016.

[50] A. Chakraborty, D. B. Roy, and D. Mukhopadhyay, "Power analysis attacks on STT-MRAM based cryptographic implementations", in *Cryptographic Hardware and Embedded Systems (CHES)*, Springer, 2017, pp. 167–187.

[51] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain", *arXiv preprint arXiv:1708.06733*, 2017.

[52] M. Imani, A. Rahimi, D. Kong, T. Rosing, and J. M. Rabaey, "Voicehd: Hyperdimensional computing for efficient speech recognition", in *Proceedings of the IEEE International Conference on Rebooting Computing (ICRC)*, IEEE, 2017, pp. 1–8.

[53] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning", in *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2017, pp. 506–519.

[54] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models", in *IEEE Symposium on Security and Privacy (S&P)*, 2017, pp. 3–18.

[55] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey", *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[56] Y. Umuroglu et al., "Finn: A framework for fast, scalable binarized neural network inference", in *International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2017, pp. 65–74.

[57] *7 series fpgas configurable logic block*, UG474, Carry chain and CLB details, Xilinx, 2018.

[58] J. Ahn, S. M. Seyedzadeh, R. Karri, and S. K. Gupta, "Security analysis of emerging nonvolatile memories: Case study on STT-RAM", in *Proceedings of the 55th Annual Design Automation Conference (DAC)*, 2018, pp. 1–6.

[59] S. Ambrogio et al., “Equivalent-accuracy accelerated neural-network training using analogue memory”, *Nature*, vol. 558, no. 7708, pp. 60–67, 2018.

[60] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, “Practical fault attack on deep neural networks”, in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2204–2206.

[61] Z. Fang, P. Zhou, and W. Zhao, “Reliability issues of STT-MRAM and prospective solutions: A review”, *IEEE Transactions on Device and Materials Reliability*, vol. 18, no. 1, pp. 3–15, 2018.

[62] I. Giechaskiel, K. B. Rasmussen, and S. Bhasin, “Leaky wires: Information leakage and covert communication between fpga soc cores”, in *ACM AsiaCCS*, 2018, pp. 15–27.

[63] D. Ielmini and H.-S. P. Wong, “In-memory computing with resistive switching devices”, *Nature Electronics*, vol. 1, no. 6, pp. 333–343, 2018.

[64] B. Krautter, R. Gnad, and M. B. Tahoori, “Fpga side channels without physical access: Remote power side-channel attacks on shared fpgas”, in *Design, Automation & Test in Europe (DATE)*, 2018, pp. 1–6.

[65] C. Kudera, M. Kammerstetter, M. Müllner, D. Burian, and W. Kastner, “Design and implementation of a negative voltage fault injection attack prototype”, in *2018 IEEE International Workshop on PAINE*, 2018, pp. 1–6.

[66] Y. Liu et al., “Trojaning attack on neural networks”, *arXiv preprint arXiv:1712.06733*, 2018.

[67] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, “Remote inter-chip power analysis side-channel attacks at board-level”, in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, 2018, pp. 1–7.

[68] X. Wang, Y. Luo, S. Sugahara, S. Fukami, and H. Ohno, “Reliability of STT-MRAM under environmental stress”, in *IEEE International Reliability Physics Symposium (IRPS)*, 2018, 2B.3-1–2B.3-6.

[69] L. Zhang et al., “Persistent fault analysis on block ciphers”, in *IACR International Conference on Fast Software Encryption*, Springer, 2018, pp. 202–222.

[70] A. Ankit and et al., “Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference”, *ACM Transactions on Architecture and Code Optimization*, vol. 16, no. 4, pp. 1–28, 2019.

[71] M. Barrera, L. Goubin, A. Fournier, and J.-B. Rigaud, “Magnetic fault injection attacks on STT-MRAM in 40 nm technology”, in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2019, pp. 151–158.

[72] L. Y. Foo, “Power delivery network step load response and 1st droop formulation”, in *EDAPS*, 2019. DOI: `10.1109/EDAPS47854.2019.9011667`.

[73] M. Imani, S. Salamat, S. Gupta, J. Huang, and T. Rosing, “Fach: Fpga-based acceleration of hyperdimensional computing by reducing computational complexity”, in *Proceedings of the 24th ASP-DAC*, 2019, pp. 493–498.

[74] K. Roy, A. Jaiswal, and P. Panda, “Towards spike-based machine intelligence with neuromorphic computing”, *Nature*, vol. 575, pp. 607–617, 2019.

[75] S. Salamat, M. Imani, B. Khaleghi, and T. Rosing, “F5-hd: Fast flexible fpga-based framework for refreshing hyperdimensional computing”, in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019, pp. 53–62.

[76] L. Goubin, M. Barrera, J.-B. Rigaud, and A. Fournier, “Fault attacks on commodity MRAM memories”, in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020, pp. 75–82.

[77] W. Liu, C.-H. Chang, F. Zhang, and X. Lou, “Imperceptible misclassification attack on deep learning accelerator by glitch injection”, in *2020 57th ACM/IEEE DAC*, IEEE, 2020, pp. 1–6.

[78] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, “Memory devices and applications for in-memory computing”, *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[79] Y. Shen et al., “Statistical persistent fault analysis of aes”, *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020.

[80] S. Wiefels, C. Bengel, N. Kopperberg, K. Zhang, R. Waser, and S. Menzel, “Hrs instability in oxide-based bipolar resistive switching cells”, *IEEE Transactions on Electron Devices*, no. 10, 2020.

[81] F. Yang and S. Ren, “Adversarial attacks on brain-inspired hyperdimensional computing-based classifiers”, *arXiv preprint arXiv:2006.05594*, 2020.

[82] X. Zhang and et al., “Statistical persistent fault analysis”, *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020, SPFA methodology with robustness to messy/multi-byte faults.

[83] G. Dessouky, A.-R. Sadeghi, and S. Zeitouni, “Sok: Secure fpga multi-tenancy in the cloud: Challenges and opportunities”, in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2021, pp. 487–506.

[84] A. Hernández-Cano, R. Cammarota, and M. Imani, “Prid: Model inversion privacy attacks in hyperdimensional learning systems”, in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2021, pp. 553–558.

[85] T. Oyama, S. Okura, K. Yoshida, and T. Fujino, “Backdoor attack on deep neural networks triggered by fault injection attack on image sensor interface”, in *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security*, 2021, pp. 63–72.

[86] J. M. Real, W. Zhao, and L. Bossuet, “Physical side-channel attacks on deep neural networks: A survey”, *Applied Sciences*, vol. 11, no. 16, p. 7369, 2021.

[87] J. Van den Herrewegen, D. Oswald, F. D. Garcia, and Q. Temeiza, “Fill your boots: Enhanced embedded bootloader exploits via fault injection and binary analysis”, *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 56–81, 2021.

[88] D. Kleyko et al., “Hyperdimensional computing: A survey”, *Journal of Artificial Intelligence Research*, vol. 75, pp. 1–64, 2022.

[89] D. Kleyko, A. Rahimi, J. M. Rabaey, and et al., “Hyperdimensional computing: A survey and taxonomy”, *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–55, 2022.

[90] Z. Zou, H. Alimohamadi, Y. Kim, M. H. Najafi, N. Srinivasa, and M. Imani, “Eventhd: Robust and efficient hyperdimensional learning with neuromorphic sensor”, *Frontiers in Neuroscience*, vol. 16, p. 858 329, 2022.

[91] H. Chen, H. E. Barkam, and M. Imani, “Hardware-optimized hyperdimensional computing for real-time learning”, in *2023 IEEE 66th International MWSCAS*, IEEE, 2023, pp. 811–815.

[92] C. Drewes et al., “Turn on, tune in, listen up: Maximizing side-channel recovery in time-to-digital converters”, in *Proceedings of the 2023 ACM/SIGDA ISFPGA*, 2023, pp. 111–122.

[93] J. Krautter, P. R. Genssler, G. Sepanta, H. Amrouch, and M. Tahoori, “Stress-resiliency of ai implementations on fpgas”, in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, IEEE, 2023, pp. 333–338.

[94] F. Liu, H. Li, Y. Chen, T. Yang, and L. Jiang, “Hyperattack: An efficient attack framework for hyperdimensional computing”, in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2023, pp. 1–6.

[95] R. Nagarajan, S. Kumar, S. Banerjee, and D. Mukhopadhyay, "Scann: Side-channel analysis of spiking neural networks", *Chips*, vol. 3, no. 3, pp. 335–349, 2023.

[96] B. Sapui et al., "Power side-channel attacks and countermeasures on computation-in-memory architectures and technologies", in *2023 ETS*, IEEE, 2023, pp. 1–6.

[97] A. Lebanov et al., "Flexible unipolar igzo transistor-based integrate and fire neurons for spiking neuromorphic applications", *IEEE Transactions on Biomedical Circuits and Systems*, vol. 18, no. 1, pp. 200–214, 2024. DOI: 10.1109/TBCAS.2023.3321506.

[98] M. Lopez et al., "A tunable multi-timescale indium-gallium-zinc-oxide thin-film transistor neuron towards hybrid solutions for spiking neuromorphic applications", *Communications Engineering*, vol. 3, Jul. 2024.

[99] X. Ma, M. Xu, Q. Li, Y. Li, A. Zhou, and S. Wang, "Dynamic workload scheduling in edge computing", in *5G Edge Computing: Technologies, Applications and Future Visions*, Springer, 2024, pp. 81–105.

[100] M. Mejri, C. Amarnath, and A. Chatterjee, "Error resilient hyperdimensional computing using hypervector encoding and cross-clustering", in *2024 IEEE 42nd VLSI Test Symposium (VTS)*, IEEE, 2024, pp. 1–7.

[101] K. Myny et al., "Flexible unipolar igzo transistor-based integrate-and-fire neurons for spiking neuromorphic applications", *IEEE Transactions on Biomedical Circuits and Systems*, vol. 18, no. 1, pp. 200–214, 2024.

[102] P. Pal et al., "Analog printed spiking neuromorphic circuit", in *IEEE DATE*, (Valencia, Spanien, Mar. 25–27, 2024), 2024, 6 S.

[103] B. Sapui and M. B. Tahoori, "Side-channel attack with fault analysis on memristor-based computation-in-memory", in *Proc. IOLTS*, 2024.

[104] B. Sapui and M. B. Tahoori, "Power side-channel analysis and mitigation for neural network accelerators based on memristive crossbars", in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE, 2024, pp. 612–617.

[105] B. Sapui et al., "Collide & conquer: Side-channel attack on hyper-dimensional computing (hdc) accelerators", in *IEEE Asian Test Symposium (ATS)*, 2025.

[106] B. Sapui et al., "Leaks beyond bits: Deep learning assisted side-channel analysis on fpga-based hyperdimensional computing", in *IEEE International Conference on Computer Aided Design (ICCAD)*, 2025.

[107] S. Zhang, K. Juretus, and X. Jiao, "Exploring hyperdimensional computing robustness against hardware errors", *IEEE Transactions on Computers*, 2025.

[108] B. Sapui et al., "Hyfault: Voltage-level fault injection attacks on fpga-based hyperdimensional computing accelerators", in *IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2026.

[109] B. Sapui et al., "When faults don't vanish: Persistent fault injection and key recovery on mram-backed aes", in *IEEE Design, Automation & Test in Europe Conference Exhibition (DATE)*, 2026.

Publication List

Brojogopal Sapui
Karlsruhe Institute of Technology (KIT), Germany
Email: brojogopal.sapui@kit.edu

Publications

1. **B.Sapui**, P.Pal, M. B. Tahoori," **When Faults Don't Vanish: Persistent Fault Injection and Key Recovery on MRAM-Backed AES**," *DATE, 2026.*

2. **B.Sapui**, and M. B. Tahoori, "**HyFault: Voltage-Level Fault Injection Attacks on FPGA-based Hyperdimensional Computing Accelerators**," *ASP-DAC*, 2026.

3. **B.Sapui**, and M. B. Tahoori, "**Leaks beyond Bits: Deep Learning Assisted Side-Channel Analysis on FPGA-based Hyperdimensional Computing**," *ICCAD*, 2025.

4. **B.Sapui**, M. Sadeghipourrudsari, and M. B. Tahoori, "**Collide & Conquer: Side-channel Attack on Hyper-dimensional Computing (HDC) Accelerators**," *ATS/ITC-Asia*, 2025.

5. **B.Sapui**, and M. B. Tahoori, "**Power Side-Channel Analysis and Mitigation for Neural Network Accelerators based on Memristive Crossbars**," *ASP-DAC*, 2024.

6. **B.Sapui**, *et al.*, "**Power Side-Channel Attacks and Countermeasures on Computation-in-Memory Architectures and Technologies**," *ETS*, 2023.

7. **B.Sapui**, S. Meschkov, and M. B. Tahoori, "**Side-Channel Attack with Fault Analysis on Memristor-based Computation-in-Memory**," *IOLTS*, 2024.

8. **B.Sapui** and M. B. Tahoori, "**Side-Channel Collision Attacks on Hyperdimensional Computing Based on Emerging Resistive Memories**," *ASP-DAC*, 2025.

9. **B.Sapui**, P. Pal, and M. B. Tahoori, "**Invited Paper: Side-Channel Vulnerability Analysis of Flexible Neuromorphic Circuits**," *ICCAD*, 2025.

10. P. Pal, **B.Sapui**, and M. B. Tahoori, "**Efficient Analog Error Correction for Printed Unary-Encoded Computing**," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2025.

11. H. Zhao, **B.Sapui**, and M. B. Tahoori, "**Highly-Bespoke Robust Printed Neuromorphic Circuits**," *DATE*, 2023.

12. **B.Sapui**, P.Pal, M. B. Tahoori," **FlexSpy: Side-Channel Spy Framework for Flexible Spiking Neuromorphic Hardware**," *Submitted: Under Review*