PHANTOM: Power Hammering Attack and Countermeasure on Multi-Tenant ReRAM Compute-in-Memory Accelerators

Ashish Reddy Bommana, *Graduate Student Member, IEEE*, Rajendra Bishnoi, Naghmeh Karimi, *Senior Member, IEEE*, Farshad Firouzi, and Krishnendu Chakrabarty, *Fellow, IEEE*

***Abstract*—The increasing demand for efficient and low-power deep neural network (DNN) inference has advanced the adoption of ReRAM-based compute-in-memory (CiM) accelerators, which perform computations directly within memory to reduce energy consumption and enhance throughput. However, such architectures are vulnerable to security threats, especially in a multi-tenant environment where multiple users share the same physical resources. This paper introduces a new attack model for multi-tenant ReRAM-based CiM, power hammering, that exploits the temperature sensitivity of ReRAM cells, inducing local temperature increases that lead to conductance drift and ultimately result in erroneous inference outcomes. This serves as a denial-of-service (DoS) attack, where malicious co-tenants degrade inferencing accuracy and system reliability for legitimate users in a shared environment, ultimately undermining trust and causing potential losses to the service provider. Additionally, we propose a novel strategy to counter this security vulnerability. In this technique, we focus on selectively protecting important weights with error compensation hardware. These important weights are treated as faults, and their computation is offloaded to compensation hardware. Simulation results confirm the effectiveness of the proposed method in ensuring accurate classification results even under adversarial conditions, thereby enabling secure multi-tenant inference on ReRAM-based CiM accelerators.**

I. Introduction

THE rapid growth of artificial intelligence (AI) applications across edge and cloud environments—spanning autonomous vehicles, wearable devices, and more—has created a demand for efficient, low-power deep neural network (DNN) inferencing solutions. To meet these requirements, compute-in-memory (CiM) or processing-in-memory (PIM) accelerators have emerged as promising alternatives to conventional von Neumann architectures (CPUs/GPUs), where they often encounter the memory wall bottleneck [1]. CiM architectures perform multiply-accumulate (MAC) operations directly within the memory fabric, thereby eliminating costly data transfers and improving overall efficiency. ReRAM-based CiM architectures have gained particular traction for DNN inference workloads due to their non-volatile nature, low read energy, and high density. Recent studies show that these accelerators can deploy complex DNN models efficiently, achieving higher throughput and lower latency in power-constrained AI applications [2], [3], [4], [5].

Additionally, to accommodate the growing demand for DNN inference across edge and cloud deployments, recent work has explored multi-tenant DNN inferencing as a strategy to optimize AI accelerator utilization [6], [7], [8]. In a multi-tenant environment, multiple users can deploy distinct DNN models simultaneously on the same accelerator. By partitioning hardware resources according to individual users' requirements, this approach maximizes resource utilization, making it an attractive solution for shared AI infrastructure.

Leveraging ReRAM-based CiM accelerators in these multi-tenant settings further strengthens this proposition [9]. Since parameter storage and MAC operations are integrated within the memory fabric, CiM architectures naturally simplify the process of allocating and isolating resources for different users. Moreover, their low-power operation and efficient on-chip processing directly translate into reduced overhead when serving multiple concurrent models. Although these accelerators promise significant efficiency and flexibility, their shared nature also introduces new security vulnerabilities that threaten the integrity of DNN inference. A key concern arises from the inherent temperature sensitivity of ReRAM cells, which directly affects their conductance levels and, consequently, the accuracy of analog MAC operations [10], [11], [12]. In a multi-tenant setting, a malicious tenant co-located on the same hardware can intentionally manipulate local temperature gradients to induce conductance drift, thereby degrading the inference accuracy of a victim tenant—effectively launching a denial-of-service (DoS) attack without requiring physical access. Prior examples of DoS attacks in multi-tenant cloud environments, such as those targeting FPGA platforms, include remote power hammering that induces timing errors, as well as fault injection [13], [14], [15], [16], [17].

While existing mitigation strategies for addressing thermal-induced conductance drift [11], [18], [19], [20], [21], [22]

Received 1 March 2025; revised 25 October 2025 and 17 January 2026; accepted 19 January 2026. Date of publication 23 January 2026; date of current version 4 February 2026. The work of Naghmeh Karimi and Farshad Firouzi was supported in part by the Semiconductor Research Corporation (SRC) under Award 2025-HW-3319. The associate editor coordinating the review of this article and approving it for publication was Dr. Stjepan Picek. *(Corresponding author: Ashish Reddy Bommana.)*

Ashish Reddy Bommana, Farshad Firouzi, and Krishnendu Chakrabarty are with Arizona State University, Tempe, AZ 85281 USA (e-mail: abommana@asu.edu).

Rajendra Bishnoi is with Delft University of Technology, 2628 CD Delft, The Netherlands.

Naghmeh Karimi is with the University of Maryland, Baltimore County (UMBC), Baltimore, MD 21250 USA.

Digital Object Identifier 10.1109/TIFS.2026.3657612

in ReRAM-based CiM are primarily designed for internal thermal management within a single-user context and are not well-suited for multi-tenant environments. In such settings, temperature variations arise not only from the workload of a single model but also from interactions between multiple concurrently running models, making localized mitigation insufficient.

To demonstrate and exploit this vulnerability, we introduce PHANTOM (Power Hammering Attack and Countermeasure on Multi-Tenant ReRAM Compute-in-Memory Accelerators), a novel power hammering attack model that leverages the shared nature of multi-tenant ReRAM CiM environments to degrade inference accuracy of co-located tenants, thereby mounting a denial-of-service (DoS) attack. While PHANTOM as an acronym includes our proposed countermeasure, we use it here to refer specifically to the attack model unless stated otherwise. A malicious tenant induces localized overheating by deliberately consuming excessive power within their allocated cores, causing conductance drift in neighboring ReRAM cells. This disruption leads to systematic computation errors in analog MAC operations, ultimately degrading the accuracy of the victim's DNN inference. To the best of our knowledge, this is the first thermal-based attack specifically targeting multi-tenant ReRAM-based CiM accelerators to degrade inference accuracy. The main contributions of this paper are as follows:

1) **Novel PHANTOM attack model:** We introduce PHANTOM, a novel power hammering attack- the first attack model to exploit temperature-induced conductance drift in ReRAM-based CiM multi-tenant accelerators.
2) **Selective weight protection:** To defend against PHANTOM, we propose a selective protection strategy guided by Hessian-based sensitivity analysis. Our approach preserves inference accuracy under thermal attacks by identifying the most critical DNN weights and offloading their computation to digital error compensation units.
3) **Comprehensive attack pattern analysis:** We extend our investigation to diverse resource partitioning schemes and attack patterns across both 2D and 3D ReRAM architectures, providing insights into how different multi-tenant configurations influence vulnerability.

The rest of the paper is organized as follows: Section II provides the related work on thermal-based attack in other computing plotforms and Section III provides relevant background on ReRAM CiM and multi-tenancy in CiM. Section IV describes the proposed PHANTOM model and the different configurations and attack patterns considered. Section V presents the countermeasure proposed to overcome security vulnerabilities. Section VI presents our simulation results. Section VII discusses the scalability of our approach and details of the fabricated chip. Finally, Section VIII concludes the paper.

II. Related Work on Multi-Tenant Platforms

In this section, we review the prior work that has explored thermal attacks in computing platforms, such as FPGA multi-tenant and multi-core systems, as well as prior security works

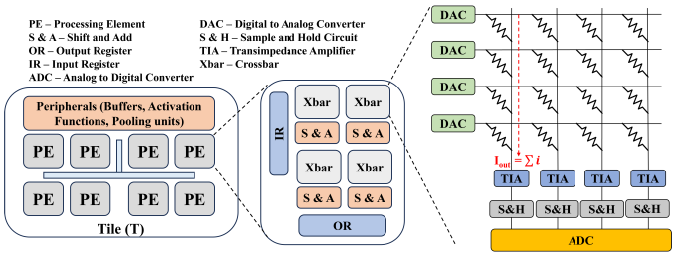


Fig. 1. CiM architecture, which includes a tile (T), processing elements (PEs), crossbars, and other peripheral circuits.

in ReRAM-based CiM. Additionally, we review prior work on mitigating thermal-induced conductance drift in ReRAM cells. Studies such as those in [23], [24], [25], [26], [27], and [28] have explored how thermal covert channel attacks can be formed, which opens up the possibility of a data breach by changing the thermal gradient across different cores in a multi-core platform. In [27], it was confirmed that the bit error rates increased to 94% due to these attacks, significantly impacting system reliability.

Similarly, vulnerabilities also exist in multi-tenant FPGA platforms, where multiple tenants share not only the same FPGA fabric but also the same Power Distribution Network (PDN). Here, an attacker can remotely exploit this shared PDN by consuming excessive power through their FPGA bit streams. This deliberate action causes a voltage drop, subsequently leading to timing faults in applications run by other tenants on the same FPGA fabric [29], [30], [31], [32], [33], [34]. Such attacks have been shown to result in DoS for legitimate users. While these works have explored temperature- or power-induced vulnerabilities in digital platforms such as multi-tenant FPGAs and multi-core platforms, our study highlights a novel threat vector unique to the shared nature of ReRAM-based CiM architectures.

III. Background

A. Compute-in-Memory Accelerators and Multi-Tenancy

Fig. 1 illustrates a typical ReRAM-based CiM architecture, which includes key components such as digital-to-analog converters (DACs), crossbar arrays, and analog-to-digital converters (ADCs) embedded within processing elements (PEs). Each PE, along with associated digital peripherals like activation units, pooling units, and input/output registers, collectively forms a functional unit known as a tile (T).

In these architectures, the crossbar arrays within each PE are responsible for executing analog matrix-vector multiplication (MVM) operations. The weight parameters are quantized and mapped as conductances in the ReRAM cells. Given the limited precision of ReRAM cells (for example, 1-2 bits per cell), a single weight is distributed among multiple cells along a row of the array. For instance, an 8-bit weight precision, combined with a 2-bit ReRAM cell precision, necessitates the use of four cells in a row. The conductance range (G_{min}, G_{max}) is divided into levels equivalent to the cell's precision, with each partial weight mapped according to its corresponding level. The architecture's DACs, which have a precision of 1-bit, process input bits in sequential clock cycles. These bits,

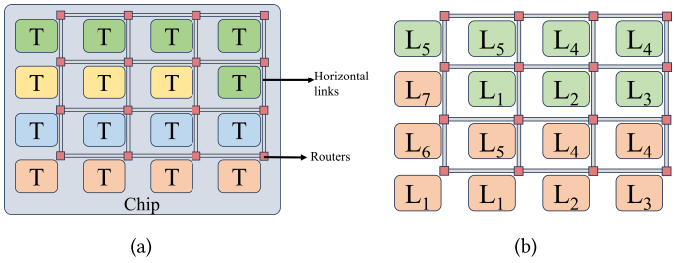


Fig. 2. (a) Multi-tenant 2D ReRAM CiM accelerator. (b) Illustration of layer-to-tile mapping for multiple tenants. L_i stands for layer i in DNN. Different colored tiles correspond to different tenants.

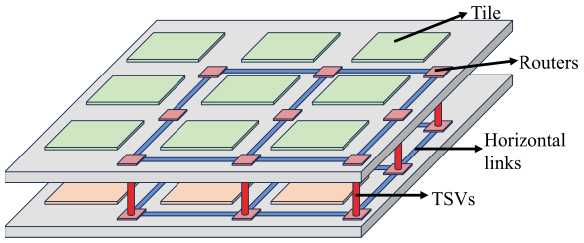


Fig. 3. Multi-tenant 3D ReRAM CiM accelerator. Different colored tiles correspond to different tenants.

applied through the DACs, induce currents that accumulate along the bit line and represent the analog output from the MVM operations within the crossbar arrays. These currents are then converted into digital values by the ADCs. Subsequent operations involve a 'shift-and-add' process, which is critical for accurately translating the bit-sliced inputs and weights MVM operations into the final digital output. However, variations in the output currents can affect the accuracy of the analog-to-digital conversion, leading to potential discrepancies in the ADC outputs. Such discrepancies can alter the computed results, deviating from the expected values and possibly resulting in misclassification.

Fig. 2–3 shows the mapping of different tenants in 2D and 3D ReRAM CiM accelerators, respectively. Fig. 3 illustrates the 3D ReRAM-based CiM system used in our evaluation. In this architecture, we assume that one tenant is exclusively mapped to one die. Through-silicon vias (TSVs) are employed to enable inter-die communication.

B. Layer-to-Tile Mapping

In CiM architectures, each DNN layer—whether convolutional or fully connected—is mapped to one or more tiles, with no two layers sharing the same tile [2], [35]. For a convolutional layer with shape (K, K, IC, OC), the 4D weight tensor is flattened into a 2D matrix of size $(K \times K \times IC, OC)$, which is then mapped onto crossbars. The number of crossbars required to map this matrix is:

$$\text{No. of Crossbars} = \left\lceil \frac{K \times K \times IC}{M} \right\rceil \times \left\lceil \frac{OC \times WP}{CP \times N} \right\rceil = N_R \times N_C$$

where $M \times N$ is the crossbar size, and WP and CP denote the weight and cell precision, respectively. Tiles consist of an array of crossbars arranged as $T_R \times T_C$, where T_R and T_C are

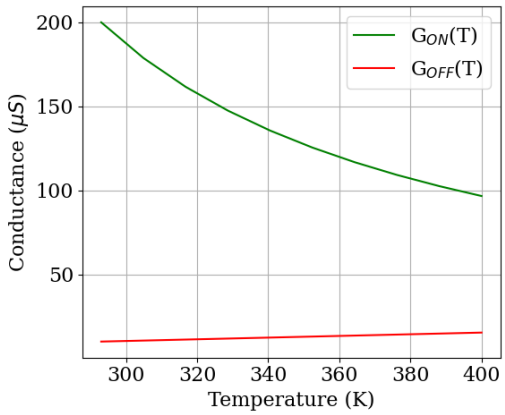


Fig. 4. Impact of temperature on off/on conductance.

the number of crossbars per tile along the row and column dimensions. The total number of tiles required per layer is given by:

$$\text{No. of Tiles} = \left\lceil \frac{N_R}{T_R} \right\rceil \times \left\lceil \frac{N_C}{T_C} \right\rceil$$

As an example, consider a layer reshaped to a 1664×10 matrix, with $WP = 8$, $CP = 2$, $T_R = 12$, and $T_C = 8$. This results in 13×1 crossbars, mapped onto 2×1 tiles. Here, the first tile hosts 12 crossbars (12.5% utilization), and the second only 1 ($\approx 1\%$ utilization). Such underutilization—common in early layers—can be mitigated by duplicating weights within underfilled tiles [35]. We adopt the layer-by-layer mapping on to the physical tiles from [35], as illustrated in Fig. 2(b), which shows tile allocation across DNN layers in multi-tenant CiM deployments.

C. Impact of Temperature on ReRAMs and DNN Inferencing

The performance of ReRAM-based CiM architectures is highly sensitive to temperature variations, which notably affect the conductance of ReRAM cells. As temperature increases, the conductance in the ON state becomes less stable, resulting in a significant decrease in the ON/OFF ratio. It has been observed that this ratio diminishes almost linearly with rising temperatures, particularly affecting the ON conductance state [36], as also evidenced by measurements from the CAIRN fabricated chip [37]. Notably, while the OFF-state conductance remains relatively stable across these temperatures, the ON-state conductance shows marked sensitivity to thermal changes.

Following [10], [11], and [36], we model a 50% linear drop in the ON/OFF ratio—from 20 at 300 K to 10 at 400 K—as illustrated in Fig. 4. The initial ON/OFF ratio of 20 is based on fabricated data from [36] for HfO_2-based ReRAM devices. Similarly, for Ta_2O_{5-x}-based fabricated ReRAM crossbar chip (CAIRN chip [37]), a 40% ON/OFF drop was observed under thermal stress from 300 K to 400 K. Further details of the fabricated chip (32×32 crossbar) and the experimental setup are provided in Section VII-A. Assuming the OFF-state conductance G_{OFF} remains invariant, the ON-state conductance $G_{\text{ON}}(T)$ can be modeled as:

$$G_{\text{ON}}(T) = (-0.1 \cdot (T - 300) + 20) \cdot G_{\text{OFF}}, \quad (1)$$

where T is the operating temperature in Kelvin. This equation captures the linear degradation in ON conductance due to thermal effects over the 300 K to 400 K range. This deviation in conductance results in distorted output currents on the bit lines. These distortions propagate through the sensing circuitry, leading to erroneous outputs from the analog-to-digital converters (ADCs), ultimately culminating in significant drops in DNN inferencing accuracy [10], [18], [19], [20], [22], [36].

IV. Phantom Attack Model

This section details the **PHANTOM** attack model, outlining its mechanisms and the various architectural configurations and attack patterns we considered to evaluate security vulnerabilities in multi-tenant ReRAM-based CiM accelerators. The model assumes a scenario where an attacker remotely degrades the inference accuracy of other tenants' DNN models by exploiting the temperature sensitivity inherent to ReRAM technology. Such an attack effectively constitutes a DoS, disrupting the normal operation of legitimate users without requiring direct access to their models. This can result in substantial consequences for service providers, including financial losses and damage to customer trust. The configurations and attack patterns described here are designed to assess the vulnerability of ReRAM-based CiM systems under diverse adversarial conditions.

A. Attack Model

The attack model assumes that the attacker has legitimate access to a designated portion of tiles within the shared ReRAM-based CiM accelerator. This access allows them to fully control these tiles to deploy their own models. This is consistent with a realistic premise in multi-tenant computing scenarios, where hardware resources are shared [9], [31], but data and model parameters remain isolated. Each tenant is assigned a dedicated set of tiles with fixed layer-to-tile mappings, and no tiles are shared across tenants.

1) Formulating Power Consumption of a Crossbar: The power consumption P of a crossbar array at each crosspoint is determined by the equation $P = VI$, where V is the voltage applied, and I is the current flowing through the cell. For a ReRAM cell acting as a resistor, the current I can be expressed as $I = V \times G$, where G represents the conductance of the cell. Consequently, the power dissipated P at a crosspoint in the crossbar is given by:

$$P_{\text{crosspoint}} = V^2 \times G \tag{2}$$

Expanding this concept to the entire array, the total power consumption P_{total} in the crossbar can be approximated by the equation:

$$P_{\text{total}} = \sum_{i=1}^{n} V_i^2 \times \left(\sum_{j=1}^{m} G_{ij} \right) \tag{3}$$

where V_i is the voltage applied to the i-th row, and G_{ij} is the conductance at the intersection of the i-th row and j-th column. This expression illustrates that total power consumption scales with both the square of input voltages and the sum of conductances (weights) along each row, emphasizing that both inputs and weights contribute to the thermal profile of the crossbar.

2) Motivation of the Attacker: Based on the Equation 3, the attacker's primary method of interference involves strategically manipulating the power consumption of their assigned tiles. By carefully selecting inputs or tuning the conductances (model weights) mapped to their tiles, they can increase local power consumption—and thus elevate temperature. This causes conductance drift in the ReRAM cells of neighboring tiles, thereby compromising the physical stability and computational accuracy of DNN models belonging to other tenants. This thermal interference reveals a subtle yet critical vulnerability in multi-tenant CiM platforms: an attacker can degrade the performance of other tenants' models without direct access, simply by manipulating their own tile's power profile. The resulting localized heating affects neighboring tiles due to the physical proximity of ReRAM tiles, leading to conductance drift and potential inference errors in victim models.

a) Attack Model Relevance: This attack model aligns with security concerns observed in prior multi-tenant systems, where shared physical infrastructure enables indirect interference across tenants. Although the attacker cannot access other tenants' data, weights, or tiles, they can still degrade the performance of neighboring models by manipulating the thermal behavior of their own tiles. This reflects realistic threats in edge and cloud deployments [9], [31], reinforcing the model's practical relevance for security assessment.

B. Attack Patterns

We explore four distinct attack patterns that exploit the temperature sensitivity of ReRAM by strategically increasing power consumption. These patterns are designed to maximize power dissipation in ReRAM-based DNN accelerators, thereby inducing localized thermal interference. The attacks arise from different combinations of two key factors: (i) the attacker's capabilities (e.g., whether they can modify input data, model weights, or both), and (ii) the defense mechanisms employed by the service provider (e.g., validation of inputs or weights). By systematically exploring these combinations, we identify four distinct classes of attack patterns. The first two patterns (A0 and A1) assume that the service provider is trusted and that tenants provide access to their DNN weights, but the service provider cannot see the input or output predictions—similar to the assumptions made in [38] and [39]. Without loss of generality, we assume that the DNN weights follow a Gaussian distribution. If the service provider monitors the distribution of programmed conductances and detects unusual deviations, the attacker is flagged.

1) Synthetic-Input Attack (A0):

a) Attack Pattern: In this attack, the adversary assumes a typical DNN model architecture—such as ResNet, VGG, DenseNet, AlexNet, etc.—mapped to the allocated tiles, without any modification to its weights. Instead, they craft adversarial inputs by setting all input values to their maximum possible level (e.g., 255 for 8-bit precision) at inference time. These artificially generated inputs are fed into the first layer to maximize row-wise voltages and hence power consumption

within the crossbars. For subsequent layers, input values are no longer under the attacker's control, as they are determined by the outputs of the previous layers and shaped by the model's internal weights.

b) Assumptions: We assume that the attacker can only manipulate the input data to the first layer and cannot modify model weights. This reflects a deployment setting where the service provider validates weight mappings to detect unusual distributions.

c) Applicability: The computational cost of this attack is negligible—it only requires setting all input values to their maximum at the first layer.

2) Synthetic-Input Attack Pattern 2 (A1):

a) Attack Pattern: Building upon A0, this attack does not assume a pre-trained DNN model. Instead, the attacker designs or trains a custom model, manipulating its architecture to ensure that inputs are maximized at each layer while maintaining weight distributions that mimic those of legitimate models (e.g., Gaussian) to avoid detection. For example, the attacker can introduce a custom activation function that adds a constant offset to the output of each layer, thereby maximizing the activations passed to the next layer.

b) Assumptions: The attacker can modify both the input data and the model weights, but avoids detection by mimicking realistic DNN weight distributions.

c) Applicability: This attack can be realized by developing a model with a customized activation function that maximizes layer-wise output activations before sending them out to the next layer.

We next consider a scenario in which the service provider validates all input data but does not verify model weights—i.e., the provider has no visibility into the tenant's model. Instead, adversarial input detectors [40], [41] are employed under the assumption that the service supports only certain workloads, such as image classification. Under this threat model, the following attack demonstrates how an adversary can exploit unchecked model weights to increase power consumption.

3) Weight-Based Attack (A2):

a) Attack Pattern: The adversary maximizes the DNN's power consumption by setting all model weights to their highest possible value (e.g., 255 for 8-bit quantization). Since power consumption in ReRAM crossbars is directly proportional to the conductance (i.e., weight magnitude), this configuration results in significantly increased power draw. To avoid detection, the attacker uses real input samples from a standard dataset such as CIFAR-10, which helps to bypass input-level validation.

b) Assumptions: The service provider performs validation on input data to detect adversarial or synthetic inputs, but does not enforce constraints or monitoring on the model weights. The attacker is assumed to have full control over the model architecture and its parameters within their allocated tiles.

c) Applicability: This attack requires no training or optimization effort. The attacker simply replaces all weights with their maximum values while using benign input samples.

We now consider the worst-case scenario, where the service provider performs no validation on either input data or model weights. Under this unrestricted setting—similar to trusted execution environments (TEEs) [42]—the attacker can simultaneously manipulate both vectors to maximize thermal interference. TEEs isolate a dedicated part of the accelerator per tenant, preventing the service provider from inspecting both inputs and weights, which enables attackers to manipulate them jointly.

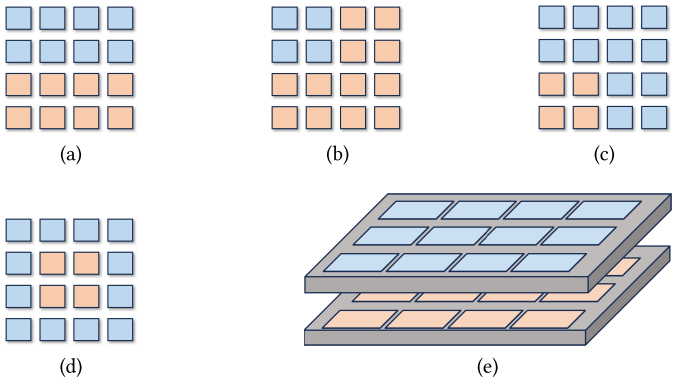


Fig. 5. Various 2D and 3D tile configurations considered in this work. In this illustration, we use a 4×4 tile configuration, black tiles represent attacker-controlled regions, while the orange tile denotes a regular user's allocation.

4) Combined Synthetic-Input and Weight-Based Attack (A3):

a) Attack Pattern: This attack combines the input maximization strategy from A1 with the weight maximization strategy from A2. The attacker sets all input values to their maximum (e.g., 255 for 8-bit inputs) and simultaneously assigns the highest possible weight values (e.g., 255 for 8-bit quantized weights) across all layers. This results in maximum voltage inputs and maximum conductance values at every crosspoint in the ReRAM crossbars, thereby maximizing instantaneous power consumption throughout the entire network.

b) Assumptions: The attacker has unrestricted control over both the input data and the model parameters, assuming a TEE model.

c) Applicability: This attack is trivial to launch, as it involves setting all weights and inputs to their maximum values without the need for any training or fine-tuning.

C. Hardware Configuration

We analyze the impact of thermal interference in two distinct hardware configurations: a 2D layout and a 3D stacked layout. In each configuration, we vary the proportion of tiles under the attacker's control and observe the resulting temperature changes and conductance drift effects. Fig. 5 illustrates the different configurations considered in this work. These configurations illustrate different scenarios in which an attacker can gain access to tiles, highlighting potential spatial vulnerabilities in multi-tenant environments. This allows us to systematically assess the attacker's ability to induce temperature-driven conductance drift and its potential to compromise inference accuracy under various architectural constraints.

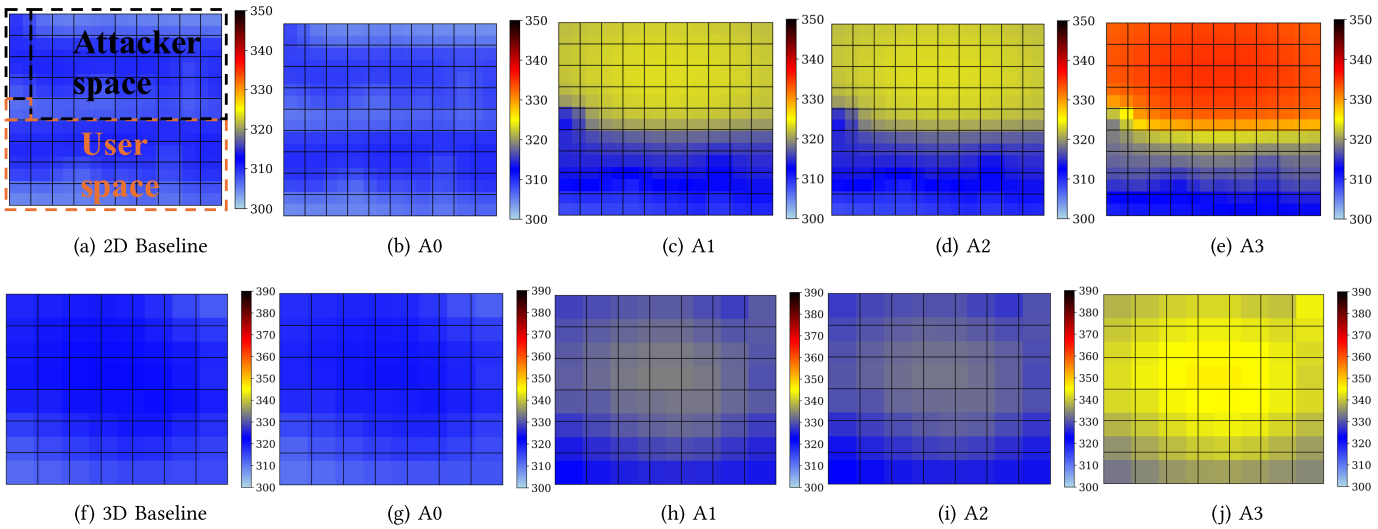


Fig. 6. Hotspot simulations illustrating temperature variations across different attack patterns for the ResNet18 model in 2D ((a)-(e)) and 3D ((f)-(j)) configurations, where the user and attacker equally share the hardware (see Fig. 5(a) and (e)).

D. Attack Results

Fig. 6 illustrates the temperature variations observed in the ResNet18 model across different attack patterns (A0 to A3) in both 2D and 3D configurations. The temperature profiles vary significantly, highlighting the impact of different attack strategies. A0 exhibits no measurable thermal disturbance, with its temperature distribution closely mirroring the baseline. This is because, in A0, we only maximized the inputs to the first layer. Although the inputs to the first layer were maximized, there is no guarantee that the subsequent layers receive maximized inputs. Thus, no significant temperature increase was observed. Consequently, A0 is omitted from further analysis as it does not contribute to performance degradation. In the 2D setup, localized heating is primarily observed in A1, A2 and A3, where attacker-controlled tiles induce sharp temperature gradients near the user's region. The severity of thermal interference increases in A3, where a more extensive heating pattern is evident, potentially leading to higher conductance drift in ReRAM cells. In contrast, the 3D setup shows a more pronounced and widespread thermal impact on the user's tiles due to tighter spatial packing. Unlike the localized effects seen in 2D, the 3D configuration exhibits a nearly uniform temperature increase, indicating that heat dissipation is more constrained in three-dimensional architectures.

V. Proposed Countermeasure

In this section, we propose a countermeasure to address the accuracy degradation caused by conductance drift resulting from elevated temperature gradients, as shown in Fig. 6. Our approach focuses on protecting critical weights within each DNN layer, recognizing that not all weights contribute equally to model performance [43]. Deviations in these critical weights can disproportionately affect overall accuracy. One potential mitigation strategy is to offload computations involving these critical weights from the analog crossbars to the digital domain. Since digital computations are not affected by temperature variations, this redirection can ensure computational integrity. However, this raises two key questions: how to identify and select critical computations for offloading, and where these computations should be redirected. To address this, we propose leveraging existing on-chip error correction units (ECUs) as digital compute fallbacks. In the remainder of this section, we present (i) the motivation for using ECUs in this context, (ii) the working of ECUs in CiM systems, and (iii) how to program them effectively to mitigate conductance drift induced by thermal interference.

A. Motivation

As illustrated in Fig. 4, ReRAM cells in low-conductance states/levels remain highly stable, exhibiting minimal drift even under elevated temperatures. This suggests that mapping computations to use low-conductance cells can enhance reliability under thermal stress. There are two general approaches to achieve this goal:

1) Complex Training-Time Conductance Shaping: One option is to train the neural network such that most weights are mapped to low-conductance states. However, this approach poses two challenges: (1) it deviates from the natural Gaussian-like weight distributions that yield high model accuracy, and (2) it introduces significant training overhead and hyperparameter tuning. Due to these limitations, we do not consider this method further.

2) Offloading Computations From Analog to Digital Domain: A more practical solution is to offload computations involving sensitive or drift-prone weights from the analog crossbars to the digital domain, using existing error correction units (ECUs). In memory systems, ECUs are typically employed to detect and correct bit errors during read operations [44], [45]. However, their role in CiM systems is fundamentally different. Unlike memory systems, which read individual words (i.e., rows), analog CiM operations perform matrix-vector multiplications (MVMs) across multiple rows and columns simultaneously. As a result, ECUs designed for memory cannot be directly applied to analog CiM

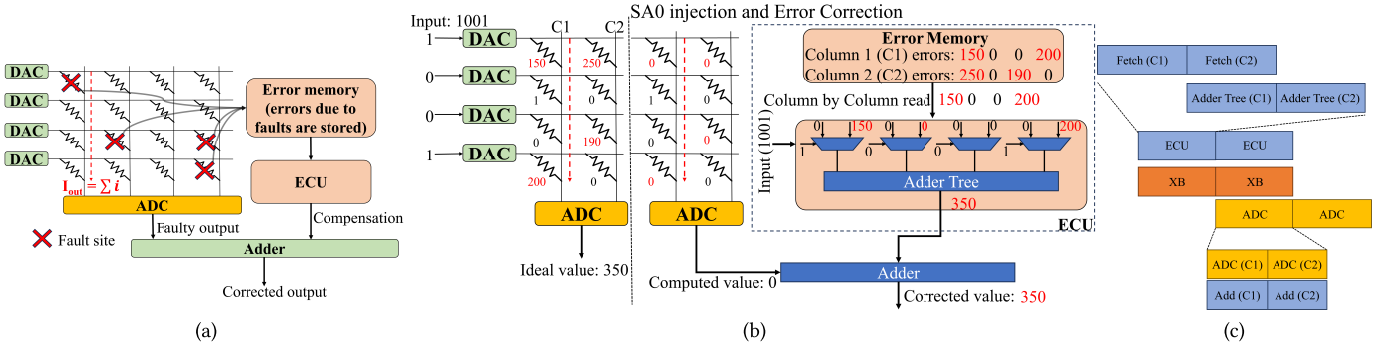


Fig. 7. (a) Overview of the error compensation circuit adapted from [46]. (b) An example of ECU for a 4×2 crossbar array. (c) the pipeline for the ECU.

architectures. References [46] and [47] introduced a specialized ECU design that enables reliable MVM computation in analog CiM systems. In our work, we adopt this ECU architecture as the foundation for thermal-resilient computation, as shown in Fig. 7(a).

The next question is how to use ECUs to mitigate conductance drift caused by thermal interference. Our aim is to perform MVM computations at low-conductance levels, which remain more stable even under high temperatures. For example, mapping weights to the minimum conductance state (e.g., zero) would yield more reliable computations. This can be viewed similarly to cells that are "stuck-at-zero (SA0)," a condition where a cell's conductance is permanently fixed at zero and normally treated as a fault. As shown in [46], such faults can be corrected using ECUs, which suggests that ECUs can also support reliable computation when weights are intentionally mapped to low-conductance states. Hence, in our approach, weights requiring protection against thermal-induced errors are treated as pseudo SA0 faults. This process necessitates the loading of only the corresponding error information for these weights into the allocated ECU error memory, as detailed in Section V-B.

For instance, a conductance level of 145 can be intentionally mapped to zero and treated as a pseudo SA0 fault. The ECU can then compensate by adding an offset of +145 to the corresponding ADC output from the crossbar to ensure reliable, correct MVM operation. However, the number of such pseudo faults that can be supported is limited by the correction budget of the ECU hardware [46]. This leads to an important question: which weights should be selected for ECU-based correction to maximize overall robustness? To address this, we identify the top-k most critical weights in each layer based on their Hessian sensitivity scores (detailed later in this section). These critical weights are treated as pseudo SA0 faults by programming them to a low-conductance state and storing their corresponding error values in the ECU's error memory. This allows the ECU to perform the necessary compensation computations during runtime (as shown in Fig. 7(b)), ensuring drift-resilient execution of the most important operations in the network.

B. Working of ECUs

Fig. 7(a) presents an overview of the ECU architecture adopted from [46], while Fig. 7(b)-(c) provides a numerical example and illustrates the ECU pipeline in the context of our proposed countermeasure. We will follow this example throughout the section to explain the ECU's operation. The left portion of Fig. 7(b) shows the initial conductance mapping of a 4×2 crossbar. Suppose that all weights above a conductance value of 150 are considered vulnerable to thermal-induced drift. These values are therefore treated as pseudo stuck-at-zero (SA0) faults, as shown in the center of the figure. The ECU corrects these errors through three key stages:

1) **Error Logging:** Errors corresponding to the faults in the crossbar are pre-logged into memory. Since weight mappings are fixed at deployment time, the errors can be computed and stored before inference begins. For instance, in Fig. 7(b), the cell at row 1, column 1 originally has a conductance value of 150. When this is treated as SA0, the error is recorded as 150. This process is repeated for the entire array, and the errors are stored in the error memory as shown in Fig. 7(b) before the inference begins.
2) **Compensation Calculation:** During inferencing, for each column, the ECU reads the logged error values and calculates compensation. The steps include: (i) fetching errors for the current column, (ii) computing the weighted errors based on the input vector using multiplexers, and (iii) summing them using an adder tree. Fig. 7(b) shows this process for column C1.
3) **Output Adjustment:** The resulting compensation values are added to the crossbar's ADC outputs to correct the MVM results. These corrections are performed by on-chip ECUs located in each PE, as described in [46].

1) ECU Pipeline Operation: As illustrated in Fig. 7, ECU computations are performed in parallel with crossbar operations. For example, during the first cycle, the crossbar (XB) computes its analog output for the input vector (e.g., 1101), while the ECU simultaneously processes the corresponding compensation. The crossbar and ECU operate on different clocks, similar to how ADCs operate [46]. We assume that the ECU clock matches the ADC clock. Like the ADC, the ECU processes data column-by-column: in one cycle, errors for column C1 are fetched; in the next, compensation for C1 is computed while errors for C2 are prepared. It is important to note that the crossbar clock is significantly slower than the ADC/ECU clock. For instance, ISAAC [2] reports crossbar operation at 10 MHz, while the ADC runs at

1.28 GHz. As shown in [46], this frequency difference allows ECU computations to complete within the crossbar clock cycle, avoiding any timing violations. As the ADC processes columns sequentially, the corresponding compensation values computed by the ECU are added to each column's output in a synchronized manner. This pipelined execution ensures that error correction is performed alongside normal crossbar operation, without introducing additional latency or disrupting throughput, as illustrated in Fig. 7.

C. Overhead of ECUs

The hardware overhead of the ECU has been reported as 12.2% in area and 10.2% in power, as per [46]. Since the ECU operates in parallel with the crossbar, it introduces no additional latency. In the example shown in Fig. 7(b), four out of eight cells in the crossbar are corrected using ECU, resulting in a 50% memory overhead. However, such a large correction budget is not practical. Prior studies [44], [45] indicate that on-die ECC memory is typically limited to correcting 10–15% of cells. Although those studies focus on memory systems (not CiM), we adopt their correction limits to model a realistic ECU budget for our system. Given this memory constraint, it is not feasible to protect all thermally sensitive weights. Therefore, it becomes essential to selectively protect only the most critical weights—those whose deviation would most impact inference accuracy. This ensures that the limited ECU resources are used efficiently to maintain computational integrity.

D. Identifying Top-k Critical Weights

As discussed in Section V-C, the limited correction budget of ECUs makes it impractical to protect all weights in the network. Therefore, it is essential to identify and protect only the most critical weights. To achieve this, we use a Hessian-based sensitivity scoring mechanism to select the top-k most critical weights per layer. This process involves two key steps:

1) Identifying Critical Weights: The criticality of each weight is determined using the diagonal elements of the Hessian matrix, as proposed in [43], [48], and [49]. The Hessian matrix captures the second-order partial derivatives of the loss function f with respect to the model parameters (weights):

$$H(w) = \begin{bmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \frac{\partial^2 f}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{bmatrix}$$

Here, w_i and w_j are weight parameters, and n is the total number of weights in the model. The diagonal entries $H_{ii}(w)$ reflect the curvature of the loss surface with respect to each individual weight w_i, enabling the determination of each weight's importance. The sensitivity score S of a weight w_i is given by [48]:

$$S = H_{ii}(w) \cdot (w_i^2) \tag{4}$$

This score quantifies the expected change in loss if the weight w_i is significantly perturbed. However, computing the full Hessian matrix for DNNs with millions of parameters is computationally expensive. To address this, we approximate and compute only the diagonal elements using Hutchinson's stochastic method from [50], which provides an efficient and accurate estimate. Prior work [48], [49], [50] demonstrates that non-diagonal elements contribute negligibly to sensitivity calculations and can be safely ignored. Moreover, these sensitivity scores need to be computed only once after the training process. Since the loss landscape is fixed at convergence, the estimated Hessian values remain valid throughout the inference process. Thus, this scoring process introduces only a one-time offline cost before deployment.

2) Protecting Critical Weights with ECU: Using the computed sensitivity scores, we select the top-k most critical weights in each layer—those whose perturbation would cause the highest loss. These selected weights are then treated as pseudo SA0 faults, and their corresponding error values are stored in the ECU's error memory prior to deployment. During inference, the ECU uses this pre-logged information to apply compensations, thereby preserving the integrity of computations involving the most critical weights. At deployment, the service provider requests an "error file" to be loaded into the ECUs - a process already followed to correct any faults in the system. At this stage, the tenant can provide their own error file corresponding to the identified pseudo-faults (i.e., critical weight compensations), which is then loaded into the ECU memory. This allows critical weight protection within a shared infrastructure.

TABLE I
KEY CIM ARCHITECTURAL SPECIFICATIONS USED IN OUR EVALUATIONS ADOPTED FROM [2] AND [35]

Parameter	Value
Crossbar clock frequency	10 MHz
Crossbar size	128×128
Number of crossbars per tile	96
Crossbar (1T-1R) power	8.0 mW per crossbar
Peripheral power	2.7 mW per crossbar
G_{ON}/G_{OFF} ratio	20:1
V_{read}	0.7 V
Power of eDRAM(P_{dram})	0.3125 mW per KB
Power of EC [46](P_{ec})	0.21 mW per crossbar
Memory overhead of EC at tile-level	10%

VI. EVALUATION AND RESULTS

A. Setup

1) DNN Models, Dataset, and Training: For our evaluation, we employed three deep neural network models—ResNet18, VGG8, and DenseNet40—trained on the CIFAR-10 dataset. All three models were trained using a learning rate of 0.01, a batch size of 128, and SGD as the optimizer.

2) Simulation Environment: All experiments were conducted on a high-performance computing node equipped with an *AMD EPYC 7302* 16-Core Processor and an *NVIDIA A5500* GPU with 24 GB of RAM.

3) CiM Architecture Specifications: We adopt a CiM approach proposed in ISAAC [2], where each tile comprises 96 crossbars. Table I provides the key architectural parameters and operating conditions, including the crossbar

clock frequency, crossbar dimensions, and power consumption estimates. Power estimates for the crossbar are derived from NeuroSim [51] (see Equation 3). The power consumption of eDRAM and peripheral circuitry within each tile is adopted from ISAAC [2], while the conductance on/off ratio is obtained from [36]. The power for the error correction unit is derived using Synopsys DC Compiler, as shown in [46].

4) Simulation Tools: We used a modified version of PytorX [52], which is based on PyTorch, to capture the impact of temperature on inferencing accuracy. Temperature profiling was performed using Hotspot [53] for both 2D and 3D configurations. For Hotspot, the ambient temperature is set to 300K. All simulations were conducted using the three DNN models on the CIFAR-10 dataset. To assess the sensitivity of weights with respect to DNN accuracy, we computed the Hessian matrix via the Hutchinson approximation as described in [50], which offers computational efficiency by avoiding direct computation of large Hessian matrices.

5) Floorplan Configurations and Attack Patterns: To investigate potential thermal interference in a multi-tenant CiM accelerator, we considered four different spacial configurations within a 2D floorplan:

1) *Under-Shared (C1)*: The attacker controls 80% of the tiles (see Fig. 5(b)).
2) *Equally-Shared (C2)*: The attacker controls 50% of the tiles (see Fig. 5(a)).
3) *Over-Shared (C3)*: The attacker controls 20% of the tiles(see Fig. 5(c)).
4) *Surrounded (C4)*: The attacker's tiles completely encircle the user's tiles, as illustrated in Fig. 5(d).

In addition to these four discrete configurations, we also varied the fraction of attacker-controlled tiles from 10% to 90% to observe how different occupancy levels influence temperature gradients and, consequently, the victim's inference accuracy. Note that the number of tiles available to the attacker is determined by the number of tiles used by the victim tenant and the selected configuration (e.g. C1/C2). For instance, under the C1 configuration, if Model A is mapped to 20 tiles, the attacker controls 80 tiles ($80/20 \cdot 20$), resulting in a total 100 tiles in the shared system. Conversely, if Model B is mapped to only 4 tiles, the attacker controls 16 tiles ($80/20 \cdot 4$), resulting in a total 20 tiles in the system. Table II provides the tile configuration for each case used in our simulations.

For the 3D floorplan, we adopted a 2-tier stacking approach, assigning the user to one tier and the attacker to the other. We examined two placements of the user's tier relative to the heat sink—either on the top (closer to heat dissipation) or at the bottom (farther from the heat dissipation interface)—to gauge how vertical proximity amplifies or mitigates thermal interference. In the *baseline* scenario, referred to as self-heating (SH), there is no malicious user. Each tenant runs a legitimate workload. The other is a normal baseline without any heating effects, including self-heating (denoted as NH). This baseline provides a reference to quantify the additional thermal stress introduced under adversarial conditions in both 2D and 3D configurations.

6) Configuration for Countermeasure: To evaluate the effectiveness of the proposed countermeasure, we configured the error correction mechanism at the tile level, with each tile capable of protecting up to 10% of its cells. This constraint follows the overhead limitations imposed by on-die memory for error correction, as outlined in [44] and [45].

TABLE II
NO. OF USER TILES (# UT), ATTACKER TILES (# AT) AND TOTAL TILES (# TT) FOR EACH CONFIGURATION

Model	# UT	Configuration							
		C1		C2		C3		C4	
		# AT	# TT	# AT	# TT	# AT	# TT	# AT	# TT
R18	41	164	205	41	82	11	52	22	63
V8	14	56	70	14	28	4	18	16	30
D40	68	272	340	68	136	17	85	31	99

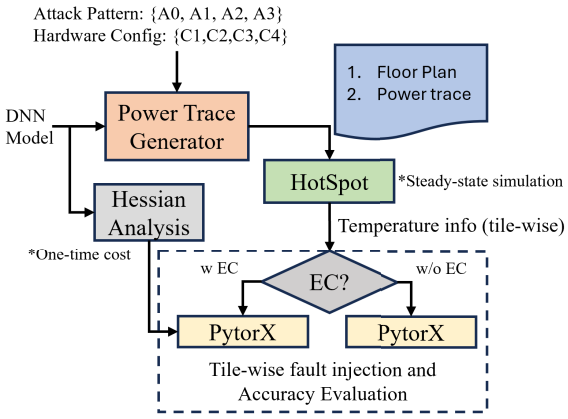


Fig. 8. Simulation framework in PHANTOM.

7) Simulation Framework: Fig. 8 illustrates the simulation framework employed in PHANTOM for evaluating accuracy under adversarial thermal conditions and assessing the effectiveness of the proposed error correction (EC) strategy. The process begins with the *Power Trace Generator*, which records dynamic power consumption across different layers and tiles during DNN inferencing. These traces, along with the floorplan description, are provided as input to HotSpot to model the spatial temperature distribution across the CiM platform. The resulting thermal map from HotSpot is then passed to PytorX, which performs tile-wise fault injection based on the temperature of each tile. When error correction is enabled, PytorX also utilizes precomputed Hessian scores to selectively protect critical weights, and the final inference accuracy is evaluated with EC applied.

B. Evaluation Results

1) Temperature Changes Observed Due to Attacks: Fig. 9 illustrates the mean, maximum, and minimum temperature variations across all user's tiles for different floorplan configurations and attack patterns in both 2D and 3D setups for three victim models: ResNet18 (R18), DenseNet40 (D40), and VGG8 (V8).

1) R18: Fig. 9 (a)-(b) present the temperature variations for ResNet18. The results indicate that among all attack patterns, the weight-input-based attack (A3) induces the most severe thermal impact on the user tiles.
2) D40: Fig. 9 (c)-(d) illustrate the temperature behavior for DenseNet40, showing a similar trend, where A3 leads to the most significant temperature increase across different configurations.

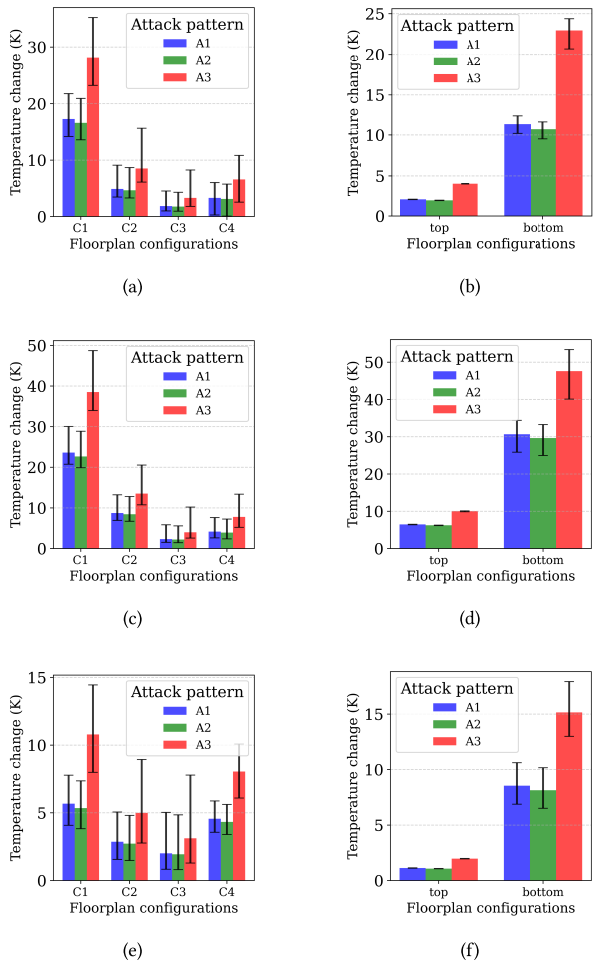


Fig. 9. Change in temperature across different attack patterns and floorplan configurations, showing average, maximum, and minimum temperature variations. (a)-(b) correspond to ResNet18, (c)-(d) represent DenseNet40, and (e)-(f) illustrate VGG8, evaluated in both 2D and 3D setups.

3) V8: Fig. 9 (e)-(f) depict results for VGG8, reinforcing the trend observed in the other models.

The slightly lower impact of A3 in the case of VGG8 (a 15K temperature change), compared to the >30K observed for ResNet18 and DenseNet40, stems from the hardware configuration used in our evaluation. From Table II, the total number of tiles in shared CiM system for VGG8 are relatively lower as compared to the other model's foot print for each configuration. This smaller system footprint inherently limits the number of tiles available to the attacker—for example, under configuration C1, the attacker controls 164 tiles in the ResNet18 setup but only 56 tiles in the VGG8 case. As a result, the total amount of heat that can be injected is lower, leading to a less pronounced temperature increase.

In the 3D setup, when user tiles are positioned near the heat sink ('top' as shown in Fig. 9(b), (d) and (f)), the attacker's thermal influence is not significant due to the heat sink being closer to the victim. Additionally, the change in temperature across all the tiles are uniform due to the vertical stacked nature, this explains the absence of distinct maximum and minimum bars in the plots as compared to the bottom case, where the victim models are placed away from the heat sink.

2) Accuracy Loss Due to Attack: To assess the functional consequences of thermal interference, we quantify the accuracy degradation induced by temperature fluctuations across three models: ResNet18 (R18), DenseNet40 (D40), and VGG8 (V8). Tables III and IV present the accuracy drop for all attack patterns and floorplan configurations in the 2D system, while Tables V and VI provide corresponding results for the 3D stacked architecture. Across both topologies, we observe a strong correlation between temperature rise and accuracy loss: higher thermal interference results in more severe inference degradation. For completeness, we also report model accuracy under self-heating (SH) and under baseline (no thermal impact) conditions.

3) Key Observations: Accuracy Degradation Patterns: Among all simulated configurations, C1A3 exhibits the most severe thermal impact across all models, which resulted in: 93.14% to 54.57% accuracy drop for R18, 86.92% to 17.76% for D40, and 85.90% to 81.61% for V8. Other configurations, such as C1A1 and C1A2, also lead to notable degradation—exceeding 10% accuracy loss for R18 and more than 60% for D40. In contrast, V8 shows minimal degradation, due to its smaller tile footprint in the shared CiM system, as shown in Table II.

In the 3D setup, vertical stacking exacerbates the thermal issue, particularly when legitimate user tiles are placed further from the heat sink. Notably, when user tiles are located closer to the heat sink, accuracy loss remains under 5% in most cases, with the exception of the D40 model under the A3-top configuration.

Overall, these findings show that the total number of tiles in a system plays a significant role in determining vulnerability to thermal interference. Specifically, configurations with fewer tiles (e.g., under 150) exhibit better thermal stability. In the 2D case, self-heating alone does not cause accuracy degradation, affirming the energy-efficient nature of CiM-based execution in benign scenarios. However, in the 3D architecture, even in the absence of malicious interference, self-heating alone can lead to a 3% drop in accuracy as seen in R18. Later, we show that this accuracy loss is also recovered with the proposed countermeasure.

C. Accuracy Recovery Through the Proposed Countermeasure

Tables III-VI also report recovered accuracy after applying our proposed countermeasure, which leverages the ECU to protect the top-10% most critical weights (see the table entries under the column 'R' for each configuration and attack pattern). We consider a 1–5% accuracy degradation to be tolerable, consistent with prior work [54], [55].

1) Highly Effective Recovery: In nearly all cases, the countermeasure restored model accuracy to within 1-2% of the baseline. This includes all cases for V8 and R18, with the exception of C1A3, and D40, excluding C1 and C2A3. These results underscore the critical role of high-sensitivity weights in inference accuracy and demonstrate that utilizing the ECU for compensating thermal-induced errors by treating critical weights as pseudo-faults (which involves storing the corresponding error info in the ECU error memory) is highly

TABLE III

ACCURACY (%) UNDER NORMAL AND ADVERSARIAL CONDITIONS IN A 2D SETUP, ILLUSTRATING ACCURACY OF DNNS WITH SELF-HEATING (SH) AND WITHOUT SELF-HEATING (NH) ACROSS DIFFERENT ATTACK PATTERNS AND CONFIGURATIONS, ACCURACY UNDER ATTACK (A) FOR EACH CASE, RECOVERED ACCURACY WITH ERROR COMPENSATION (R), AND THE CORRESPONDING ENERGY OVERHEAD (EO) OF ECU (%)

Model	NH	C1							C2							C3							EO
		SH	A1		A2		A3		SH	A1		A2		A3		SH	A1		A2		A3		
			A	R	A	R	A	R		A	R	A	R	A	R		A	R	A	R	A	R	
R18	93.98	93.06	80.89	91.92	82.02	92.09	54.57	89.47	93.44	92.65	93.73	92.71	93.75	91.74	93.47	93.72	93.5	93.8	93.5	93.8	93.40	93.70	5.8
V8	86.90	86.37	84.47	86.16	84.66	86.23	81.61	85.57	86.59	86.25	86.69	86.28	86.71	85.88	86.60	86.90	86.6	86.85	86.62	86.85	86.53	86.80	5.4
D40	86.92	86.16	26.32	62.54	27.25	64.34	17.76	36.21	86.75	81.06	85.78	81.5	85.80	67.47	83.28	86.90	86.58	86.70	86.66	86.75	86.21	86.67	6.5

TABLE IV

ACCURACY (%) AND ENERGY OVERHEAD OF ECU (%) FOR C4 IN 2D SETUP

Model	NH	C4							EO
		SH	A1		A2		A3		
			A	R	A	R	A	R	
R18	93.98	93.72	92.64	93.69	92.76	93.68	91.34	93.35	5.9
V8	86.90	86.90	85.33	86.52	85.55	86.55	83.59	85.95	5.5
D40	86.92	86.80	83.61	86.13	83.93	86.20	75.77	85.02	6.5

TABLE V

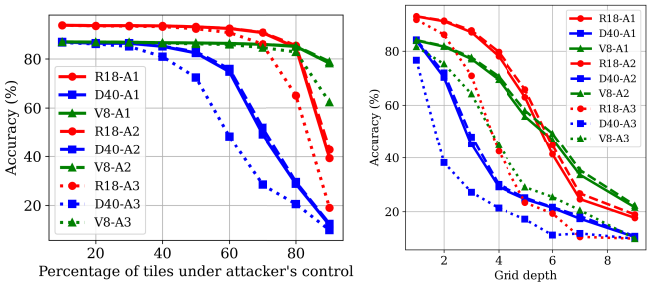
ACCURACY (%) AND ENERGY OVERHEAD OF ECU (%) WHEN THE VICTIM TENANT IS AWAY FROM THE HEAT SINK (BOTTOM) IN A 3D SETUP

Model	NH	SH	A1		A2		A3		EO
			A	R	A	R	A	R	
R18	93.98	89.98	72.79	91.27	74.4	91.40	37.47	87.08	5.8
V8	86.90	86.18	82.16	85.27	82.4	85.74	75.42	84.14	5.4
D40	86.92	86.91	22.14	51.75	22.6	54.18	13.95	28.28	6.5

TABLE VI

ACCURACY (%) AND ENERGY OVERHEAD OF ECU (%) WHEN THE VICTIM TENANT IS NEAR THE HEAT SINK (TOP) IN A 3D SETUP

Model	NH	SH	A1		A2		A3		EO
			A	R	A	R	A	R	
R18	93.98	93.92	93.66	93.88	93.66	91.89	93.47	93.70	5.8
V8	86.90	86.90	86.90	86.90	86.90	86.90	86.90	86.88	5.4
D40	86.92	86.64	86.29	51.75	84.24	86.34	76.83	84.99	6.5



(a) (b)

Fig. 10. (a) Impact of the attacker's allocated spatial resources on inferencing accuracy across all models and attack patterns. (b) Impact of grid depth on inferencing accuracy across all models and attack patterns.

effective. Furthermore, this improvement was achieved with a modest energy overhead of approximately 6%.

2) Moderately Effective Recovery: In thermally severe configurations—specifically R18 under C1A3 and D40 under C2A3—the countermeasure was moderately effective in recovering the accuracy, i.e., restoring accuracy to within 5% of the baseline (SH). In Section VII-B, we explore the use of non-uniform top-k strategies to further enhance performance in such cases.

3) Ineffective Recovery: The proposed method was ineffective in recovering D40's accuracy under the C1 configuration, where accuracy dropped far below 5% of the baseline even after correction. This failure is attributed to a substantial temperature increase exceeding 40 K (as shown in Fig. 9) and a large tile footprint—340 tiles, as reported in Table II. In comparison, the same model under A3, with only 136 tiles, showed effective recovery, highlighting the dominant role of tile footprint in determining recovery feasibility. These observations collectively indicate that to ensure thermal resilience, the total number of tiles in the system should be constrained to around 100–150.

4) Mitigation Strategies When Recovery Fails: As discussed in Section IV, all considered attack patterns (A1–A3) are realistic under different operational assumptions. Therefore, in cases where ECU-based error correction is insufficient, it becomes imperative to neutralize the threat through alternative measures. One effective approach involves integrating temperature sensors across the chip. For example, under the D40 C1A3 scenario, the thermal spike exceeds 50 K, which is detectable via localized sensors. This approach is conceptually similar to the method proposed in [56], where voltage-drop sensors are embedded across FPGA fabric to flag abnormal power activity from malicious agents.

However, it is important to strike a balance. Flagging attacks based on small temperature deviations (e.g., 1–25 K) can degrade trust in the service provider, especially if legitimate workloads are disrupted unnecessarily. To address this, we propose setting 25 K as a practical threshold for raising threat alarms. Our evaluation shows that the ECU-based countermeasure can recover accuracy effectively for temperature rises up to 25 K. This makes our approach both practical and efficient for maintaining service continuity under moderate thermal interference.

5) Observations in 3D Stack Configurations: In 3D stacked systems, the proposed countermeasure remains effective under attack patterns A1 and A2. Accuracy lost due to both self-heating and malicious interference is successfully recovered for R18 and V8. For D40, recovery is only observed when the temperature rise remains below 25 K, as illustrated in Fig. 9(d). Beyond this threshold, thermal errors overwhelm the correction capability. These results confirm that the 25 K temperature rise serves as a reliable threshold for threat-tolerant inference in both 2D and 3D systems. When thermal deviations exceed this bound, thermal sensor telemetry becomes essential for flagging the malicious user for further diagnosis.

D. Accuracy Trends Across Different Settings

To analyze the worst-case and best-case scenarios for a user, we varied the percentage of tiles controlled by the attacker from 10% to 90%, as shown in Fig. 10(a). A key observation across all models is that accuracy degradation

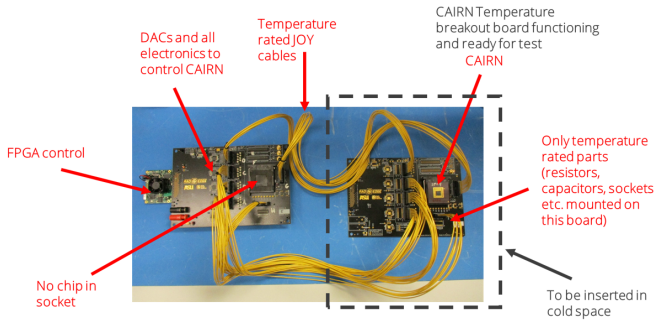


Fig. 11. Illustration of the CAIRN's hardware platform for temperature analysis.

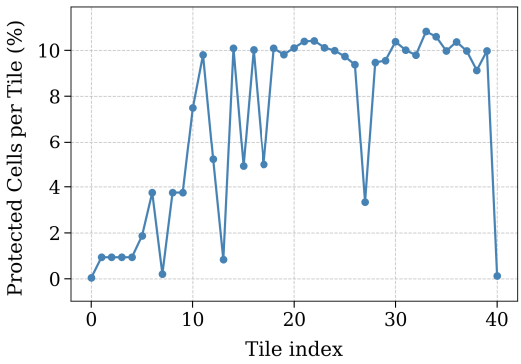


Fig. 12. Percentage of critical cells in each tile when protecting top-10% critical weights in each layer for ResNet18.

becomes significant when the attacker controls at least 50% of the tiles. This suggests that beyond a certain threshold, the attacker's influence leads to severe thermal interference, significantly impacting inference performance.

We conducted a separate experiment to evaluate how the spatial arrangement of attacker-controlled tiles surrounding the user tiles — parameterized as "grid depth" — influences accuracy degradation under thermal stress. As illustrated in Fig. 5(d), a grid depth of 1 corresponds to a 4×4 grid, where the user's tiles occupy the central 2×2 region and are directly surrounded by attacker tiles. A grid depth of 2 expands the layout to a 5×5 grid, where the same 2×2 user region is enclosed by two concentric rings of attacker-controlled tiles. Fig. 10 shows the resulting accuracy degradation as grid depth increases. The results indicate that even with relatively weak attacks (A1), significant accuracy drops occur when the user is surrounded by as few as two rings of attacker tiles. These findings have important implications for system-level resource allocation: service providers should avoid assigning users to tile regions that are completely enclosed by other tenants, as this can lead to increased thermal exposure and reliability degradation.

VII. Discussion

A. Validation Against Real Chip (CAIRN) Measurements

To validate the temperature-induced conductance drift modeled in this work, we utilized empirical conductance variation traces obtained from fabricated ReRAM crossbar used in the CAIRN experimental platform [37]; see Fig. 11. These ReRAM devices are fabricated using a TiN/Ta (15 nm)/Ta_2O_{5-x} (10 nm)/TiN material stack and operate with set/reset voltages ranging from 0.8–1.5 V and–1.0–2.5 V, respectively. The CAIRN chip features a 32×32 ReRAM-based crossbar array, integrated with peripheral circuitry in a 130 nm CMOS technology. It incorporates a 10-bit ramp ADC per column for current sensing, along with external DACs for pulse generation. The system is interfaced with an FPGA and controlled via a custom Python-based software framework.

In our validation experiment, 100 ReRAM devices were programmed to 10 distinct conductance levels ranging from 10 nS to 225 nS and subjected to temperatures from 173 K to 400 K. For the purpose of this study, we focus on the behavior between 300 K and 400 K, which is more relevant to practical on-chip thermal conditions. At 300 K, the nominal ON and OFF conductance values were 190 nS and 15 nS, respectively. At 400 K, these values shifted to 150 nS (ON) and 20 nS (OFF), resulting in a degradation of the ON/OFF ratio from 12.6 to 7.5, a 40% drop. In our simulation framework, we modeled a conductance drop of 50% in the same temperature range, which closely aligns with the observed experimental behavior. This alignment supports the validity of the conductance degradation model used in our evaluations.

B. Non-Uniform Top-k Error Correction

As shown in Table III, the proposed method demonstrates moderate effectiveness in configuration C2A3 (for D40). Hence, to further improve the performance of the countermeasure, we leverage a non-uniform top-k strategy across different layers for an overall system resilience. Consider a scenario where we aim to protect the top 10% most critical weights in each DNN layer, and each ECU has the capacity to correct errors in up to 10% of the total cells within a tile. In our system, each tile consists of 96 crossbars. Under this setup, we observe that certain tiles exceed the available ECU memory capacity, making it infeasible to fully compensate even the top 10% of layer weights. As shown in Fig. 12, this imbalance is especially evident in middle layers, where critical weights tend to cluster disproportionately within specific tiles. For example, if a layer spans multiple tiles, one tile may contain significantly more critical weights than others, thereby exceeding its individual correction budget. This challenge is exacerbated by the heterogeneous nature of DNN layers in terms of size and mapping density. Deeper layers often span more tiles and may require correction for more than 10% of a tile's cells to fully protect just the top 10% of weights (see tiles 20–23 in Fig. 12). However, the limited correction budget prevents such full-scale protection. In contrast, early layers—such as the first layer of ResNet-18 mapped to tile 0—require far less correction. This layer maps to only 2 crossbars out of the 96 available within a tile, and protecting its top 10% of critical weights requires protecting only 1–2% of the tile's cells, while the budget of ECU has the capability to correct up to 10% of tile's cells. Similar trend is observed for the last layer which also requires only 3-5 crossbars. This stark variation in per-layer protection demand motivates a heterogeneous top-k allocation strategy

TABLE VII
PERFORMANCE OF THE COUNTERMEASURE ON CIFAR-100 DATASET

Config	NH	SH	A1		A2		A3	
			A	R	A	R	A	R
C1	78.74	75.71	43.59	63.74	45.87	64.86	15.23	41.7
C2		77.13	74.74	76.63	74.51	76.77	71.21	75.21
C3		77.94	77.4	78.21	77.43	78.24	76.73	77.85
C4		77.35	74.56	76.68	74.71	76.75	70.36	74.82
top		78.73	78.4	78.62	78.43	78.65	77.53	78.32
bottom		65.69	30.69	56.37	33.03	57.79	7.8	28.54

that adjusts the protection threshold based on each layer's mapping density and critical weight distribution.

For example, under a fixed top-10% strategy, the critical weights in layer A may occupy only 2% of the cells in a tile, while layer B may require correction for 12% of tile cells, exceeding the ECU's 10% capacity for that tile. To improve protection efficiency, we can increase the top-k for layer A (e.g., protect top-30%) and reduce it for layer B (e.g., protect top-8%). Since ECU operations rely only on pre-stored error data, this layer-specific variation is straightforward to implement. For tiles that exceed the memory threshold, the fraction of protected weights can be reduced accordingly, ensuring compliance with hardware constraints.

With top-30 settings where feasible for different layers in D40, the C2A3 accuracy of D40 improves from 83.28% to 85.33%, with an additional energy overhead of only 0.5%, demonstrating the effectiveness of the proposed non-uniform top-k approach.

C. Performance of ECU on CIFAR-100

Table VII presents the performance of the proposed ECU-based countermeasure on the CIFAR-100 dataset. For configurations C2, C3, and C4, the ECU remains effective, successfully restoring accuracy across all attack patterns. This demonstrates that the countermeasure continues to perform reliably even under more complex workloads.

Compared to CIFAR-10, however, recovery on CIFAR-100 is more challenging due to the larger number of classes and greater semantic overlap between them, which reduces the model's inherent error tolerance under perturbations. This limitation becomes evident in configurations such as C1 and bottom, where accuracy under attack drops significantly and recovery is constrained. In C1, the attacker is mapped to 164 out of 205 tiles, leading to a high degree of thermal interference and reduced recovery. As observed in Section VI-B.4, keeping the total number of active tiles below 150 helps limit thermal impact. This design choice contributes to the successful recovery observed in C2, C3, and C4—where the total tile count was under this threshold—highlighting that effective thermal provisioning complements ECU performance, even for complex datasets. In the 3D setup (bottom), thermal coupling between attacker and victim further amplifies the interference, resulting in more severe degradation. This suggests that vertically stacked CiM architectures may pose additional challenges for secure multi-tenant deployments.

Overall, these results underscore the importance of intelligent workload mapping and thermal-aware resource allocation in CiM systems. As future work, we aim to explore adaptive mapping and resource-allocation strategies for reliable and secure CiM.

D. Comparison With Prior Fault Injection Attacks and Mitigations in CiM

1) Prior Work on Fault-Injection Attacks in CiM: In contrast to our approach, prior fault-injection and attack models—such as side-channel [57], [58], [59], [60] and write-hammer attacks [61], [62], and row-hammer attacks—are fundamentally different in both mechanism and threat scope. Side-channel attacks exploit physical signal leakages (e.g., power, timing, or electromagnetic emissions) to infer sensitive information about the data or model parameters, but they still require close physical proximity or specialized measurement setups. Row-Hammer and Write-Hammer attacks, on the other hand, intentionally overstress specific rows or wordlines through repeated write operations to induce physical faults such as bit flips or stuck-at errors. These attacks directly manipulate device reliability and also assume low-level access to the hardware.

Our proposed attack model differs significantly—it does not rely on physical interference or direct fault injection. Instead, it leverages workload-driven power manipulation during normal inference operations to induce localized thermal gradients. This creates a remote, non-invasive pathway to degrade model accuracy without physically tampering with the chip or altering stored data.

2) Prior Work on Fault Mitigation Under Thermal Stress in CiM: Several techniques have been proposed to mitigate the impact of thermal-induced conductance drift and variability in ReRAM-based CiM architectures [11], [18], [19], [20], [21], [22]. A common class of methods focuses on spatial remapping—such as tile, layer, or row remapping—to distribute computation across regions with different thermal profiles [11], [18], [19], [20], [21]. These approaches dynamically reassign high-activity regions to cooler zones to reduce self-heating and maintain a more uniform temperature distribution across the array. However, they primarily target self-induced thermal imbalance within a single-tenant context and lack mechanisms to counter external interference or thermally coupled disturbances arising from co-located tenants in multi-tenant deployments.

Variation-aware and drift-aware retraining schemes have been proposed to adapt model weights to device-level non-idealities and gradual conductance drift [63], [64]. These approaches typically involve fine-tuning or full retraining of large models using updated device statistics or synthetic drift models. While effective in controlled offline settings, such retraining demands significant computational resources and access to the original training dataset—requirements that are often impractical or infeasible for deployed systems, especially in cloud or multi-tenant scenarios where the hardware is only accessed at inference-time.

Overall, prior mitigation strategies primarily focus on single-user reliability or offline adaptation, and do not directly address runtime fault resilience or localized thermal interference introduced by untrusted tenants. Our proposed

TABLE VIII
ENERGY OVERHEAD (%) OF ECU AS THE SPEED-UP FACTOR IS INCREASED

Speed-up Factor	DenseNet40	ResNet18	VGG8
1	6.5	6.0	5.5
2	6.9	6.4	6.0
4	8.4	7.3	6.8
8	14.3	10.2	9.0
16	38.2	11.4	9.8
32	85.9	11.8	10.3

ECU-based countermeasure is therefore complementary—it operates online during inference, enabling localized recovery without requiring model retraining or access to the original dataset. Furthermore, since the proposed approach leverages the existing ECU functionality to correct errors in critical operations by utilizing the available capacity of the ECU error memory (to store error values for weights treated as pseudo-faults), it incurs no additional area or timing overhead beyond the standard ECU implementation, making it a practical and hardware-efficient solution for multi-tenant CiM systems.

E. The Impact of Speed-up on Error Correction

In our previous evaluations, we considered the impact of varying the critical weight protection fraction (top-k) without accounting for potential speed-ups in processing. However, many layers in the evaluated DNN models, particularly early layers like layer-1, are significantly underutilized. For instance, layer-1 requires only 2 crossbars, while each tile contains 96 crossbars. This underutilization enables duplication of weights within tiles to improve throughput. With duplication, we can process multiple input strides in parallel. For example, if layer-1 needs to process 1000 input strides, and its weights are duplicated 48 times within a tile, then 48 strides can be processed simultaneously. To evaluate how such speed-ups affect our proposed error correction mechanism, we introduced different speed-up configurations into our experiments, as shown in Table VIII. These results help quantify the trade-offs between accelerated execution and the effectiveness of ECU. It is important to note that a speed-up of 32x does not imply that all layers can achieve this level of efficiency. The actual speed-up is contingent on the layer's utilization within a tile. If a layer occupies more than 50% of the crossbars, no speed-up will occur for that layer. Our observations indicate that most larger layers do not achieve significant speed-up.

From the table, it is evident that the energy overhead increases significantly with the introduction of speed-up. This rise in energy consumption is attributed to the need to duplicate weights onto multiple crossbars, requiring the EC to protect the critical weights in those duplicated crossbars. Consequently, the increase in compensation mechanisms necessary to safeguard these weights escalates the energy overhead. For this specific experiment, we maintained the top-k at 10%. Additionally, despite the increase in speed-up, we did not observe any significant changes in the accuracy of the models. This stability is primarily because the speed-up predominantly affects smaller layers, where even with an increased number of weights to protect (including duplicated ones), the ECU was able to maintain performance within its error correction capabilities, even at a 32x speed-up. This observation highlights that while smaller layers benefit from speed-up without compromising accuracy, they introduce additional energy costs that need to be carefully managed.

VIII. CONCLUSION

This study is the first to explore thermal-based attacks in multi-tenant ReRAM CiM accelerators. We have analyzed various configurations and attack patterns, showing that thermal interference can raise temperatures by 30–50 K, leading to significant accuracy degradation across both 2D and 3D setups. To address this, we have proposed an error compensation strategy that identifies critical weights and offloads their computation to thermally stable digital units. This selective protection mitigates temperature-induced faults within CiM. In addition, we have discussed CiM design constraints—specifically, the number of tiles—that can help reduce thermal interference in multi-tenant scenarios. Simulation results have showed that, with the proposed countermeasure, the accuracy loss in models such as ResNet18, VGG8, and DenseNet40 remains within 5% when the temperature rise under attack is limited to 25 K, with energy overheads kept below 10%. These findings underscore the effectiveness and practicality of the proposed EC strategy in enhancing the resilience of CiM systems against power hammering attacks.

ACKNOWLEDGMENT

The authors would like to thank Prof. Matthew Marinella and his students Alex Dozier and Ryan Dempsey from Arizona State University, Tempe, AZ, USA, and Sandia National Laboratories, Albuquerque, NM, USA, for providing experimental device-variation data under thermal stress from fabricated ReRAM cells, which were used to validate the simulation setup in this work.

REFERENCES

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[2] A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 14–26.

[3] X. Yang, B. Yan, H. Li, and Y. Chen, "ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2020, pp. 1–9.

[4] S. Huai et al., "CRIMP: Compact & reliable DNN inference on in-memory processing via crossbar-aligned compression and nonideality adaptation," *ACM Trans. Embedded Comput. Syst.*, vol. 22, no. 5, pp. 1–25, Oct. 2023.

[5] A. R. Bommana, F. Firouzi, and K. Chakrabarty, "COMET-3D: Compute-in-memory based transformer accelerator with optimized pipeline and 3D heterogeneous integration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Aug. 21, 2025, doi: 10.1109/TCAD.2025.3601526.

[6] S. Kim, J. Zhao, K. Asanovic, B. Nikolic, and Y. S. Shao, "AuRORA: Virtualized accelerator orchestration for multi-tenant workloads," in *Proc. 56th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2023, pp. 62–76.

[7] A. Karatzas and I. Anagnostopoulos, "Balancing throughput and fair execution of multi-DNN workloads on heterogeneous embedded devices," *IEEE Trans. Emerg. Topics Comput.*, vol. 13, no. 2, pp. 409–422, Apr. 2025.

[8] S. Zeng et al., "Serving multi-DNN workloads on FPGAs: A coordinated architecture, scheduling, and mapping perspective," *IEEE Trans. Comput.*, vol. 72, no. 5, pp. 1314–1328, May 2023.

[9] B. Li, D. Zhong, X. Chen, and C. Liu, "A collaborative PIM computing optimization framework for multi-tenant DNN," 2024, *arXiv:2408.04812*.

[10] H. Shin, M. Kang, and L.-S. Kim, "A thermal-aware optimization framework for ReRAM-based deep neural network acceleration," in *Proc. 39th Int. Conf. Computer-Aided Design*, Nov. 2020, pp. 1–9.

[11] M. V. Beigi and G. Memik, "Thermal-aware optimizations of ReRAM-based neuromorphic computing systems," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf.*, 2018, pp. 1–6.

[12] K. Smagulova, M. E. Fouda, and A. Eltawil, "Thermal heating in ReRAM crossbar arrays: Challenges and solutions," *IEEE Open J. Circuits Syst.*, vol. 5, pp. 28–41, 2024.

[13] S. Tang, J. Wang, Z. Chen, and S. Guo, "A covert and efficient attack on FPGA cloud based on adaptive RON," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 5, pp. 4641–4653, Sep. 2025.

[14] H. Nassar et al., "DoS-FPGA: Denial of service on cloud fpgas via coordinated power hammering," in *Proc. 43rd IEEE/ACM Int. Conf. Comput.-Aided Design*, New York, NY, USA, 2025, pp. 1–9.

[15] T. Chen, Y.-A. Tan, W. Li, Z. Ci, and N. Shi, "Toward secure program execution in multi-tenant cloud FPGA environments," *J. Supercomput.*, vol. 81, no. 8, p. 871, May 2025.

[16] H. Nassar, J. Krautter, L. Bauer, D. Gnad, M. Tahoori, and J. Henkel, "Meta-scanner: Detecting fault attacks via scanning FPGA designs metadata," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 11, pp. 3443–3454, Nov. 2024.

[17] F. Zhang, Z. Wang, H. Shen, B. Yang, Q. Wu, and K. Ren, "DARPT: Defense against remote physical attack based on TDC in multi-tenant scenario," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 559–564.

[18] G. Narang, C. Ogbogu, J. R. Doppa, and P. P. Pande, "TEFLON: Thermally efficient dataflow-aware 3D NoC for accelerating CNN inferencing on manycore PIM architectures," *ACM Trans. Embedded Comput. Syst.*, vol. 23, no. 5, pp. 1–23, Aug. 2024.

[19] P.-Y. Chen, F.-Y. Gu, Y.-H. Huang, and I.-C. Lin, "WRAP: Weight RemApping and processing in RRAM-based neural network accelerators considering thermal effect," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2022, pp. 1245–1250.

[20] B. K. Joardar, J. R. Doppa, P. P. Pande, H. Li, and K. Chakrabarty, "AccuReD: High accuracy training of CNNs on ReRAM/GPU heterogeneous 3-D architecture," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 5, pp. 971–984, May 2021.

[21] X. Liu, M. Zhou, T. S. Rosing, and J. Zhao, "HR3AM: A heat resilient design for RRAM-based neuromorphic computing," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.

[22] J. Meng et al., "Temperature-resilient RRAM-based in-memory computing for DNN inference," *IEEE Micro*, vol. 42, no. 1, pp. 89–98, Jan. 2022.

[23] Z. Shao, M. A. Islam, and S. Ren, "Heat behind the meter: A hidden threat of thermal attacks in edge colocation data centers," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2021, pp. 318–331.

[24] P. Rahimi, A. K. Singh, and X. Wang, "Fan speed control based defence for thermal covert channel attacks in multi-core systems," in *Proc. 29th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Oct. 2022, pp. 1–4.

[25] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun, "Thermal covert channels on multi-core platforms," in *Proc. 24th USENIX Conf. Secur. Symp.*, 2015, pp. 865–880.

[26] H. Huang, X. Wang, Y. Jiang, A. K. Singh, M. Yang, and L. Huang, "Detection of and countermeasure against thermal covert channel in many-core systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 2, pp. 252–265, Feb. 2022.

[27] P. Rahimi, A. K. Singh, and X. Wang, "Selective noise based power-efficient and effective countermeasure against thermal covert channel attacks in multi-core systems," *J. Low Power Electron. Appl.*, vol. 12, no. 2, p. 25, May 2022.

[28] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici, "PowerHammer: Exfiltrating data from air-gapped computers through power lines," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1879–1890, 2020.

[29] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2018, no. 3, pp. 44–68, Aug. 2018.

[30] D. R. E. Gnad, V. Meyers, N. M. Dang, F. Schellenberg, A. Moradi, and M. B. Tahoori, "Stealthy logic misuse for power analysis attacks in multi-tenant FPGAs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 1012–1015.

[31] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 3, pp. 1–26, Sep. 2019.

[32] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," *IEEE Design Test*, vol. 38, no. 3, pp. 58–66, Jun. 2021.

[33] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "Remote fault attacks in multitenant cloud FPGAs," *IEEE Design Test*, vol. 39, no. 4, pp. 33–40, Aug. 2022.

[34] G. Dessouky, A.-R. Sadeghi, and S. Zeitouni, "SoK: Secure FPGA multi-tenancy in the cloud: Challenges and opportunities," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Sep. 2021, pp. 487–506.

[35] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *IEDM Tech. Dig.*, Dec. 2019, pp. 32.5.1–32.5.4.

[36] C. Walczyk et al., "Impact of temperature on the resistive switching behavior of embedded HfO_2-based RRAM devices," *IEEE Trans. Electron Devices*, vol. 58, no. 9, pp. 3124–3131, Sep. 2011.

[37] D. Wilson et al., "A discovery platform to characterize emerging nonvolatile memories for computing," in *Proc. IEEE 42nd VLSI Test Symp. (VTS)*, Apr. 2024, pp. 1–5.

[38] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1651–1669.

[39] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.

[40] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–11.

[41] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 289–303.

[42] F. Tramèr and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 9360–9378.

[43] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, "PyHessian: Neural networks through the lens of the Hessian," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 581–590.

[44] K. C. Chun et al., "A 16-GB 640-GB/s HBM2E DRAM with a data-bus window extension technique and a synergetic on-die ECC scheme," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 199–211, Jan. 2021.

[45] Y. Ryu et al., "A 16 GB 1024 GB/s HBM3 DRAM with source-synchronized bus design and on-die error control scheme for enhanced RAS features," *IEEE J. Solid-State Circuits*, vol. 58, no. 4, pp. 1051–1061, Apr. 2023.

[46] A. R. Bommana et al., "SEC-CiM: Selective error compensation for ReRAM-based compute-in-memory," in *Proc. IEEE Int. Test Conf. (ITC)*, Nov. 2024, pp. 177–186.

[47] A. R. Bommana et al., "DEAR: Dependable 3D architecture for robust DNN training," in *Proc. Design, Autom. Test Eur. Conf. (DATE)*, Mar. 2025, pp. 1–7.

[48] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2, 1989, pp. 598–605.

[49] Z. Yan, X. S. Hu, and Y. Shi, "SWIM: Selective write-verify for computing-in-memory neural accelerators," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 277–282.

[50] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. W. Mahoney, "ADAHESSIAN: An adaptive second order optimizer for machine learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 10665–10673.

[51] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.

[52] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[53] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.
[54] V. Leon et al., "Approximate computing survey, part I: Terminology and software & hardware approximation techniques," *ACM Comput. Surveys*, vol. 57, no. 7, pp. 1–36, Mar. 2025.
[55] J. Park, E. Amaro, D. Mahajan, B. Thwaites, and H. Esmaeilzadeh, "AxGames: Towards crowdsourcing quality target determination in approximate computing," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 623–636, Mar. 2016.
[56] X. Yan, H. Qiu, and T. Zhang, "UniGuard: A unified hardware-oriented threat detector for FPGA-based AI accelerators," in *Proc. 34th Int. Conf. Field-Programmable Log. Appl. (FPL)*, Sep. 2024, pp. 164–170.
[57] Z. Wang, F.-H. Meng, Y. Park, J. K. Eshraghian, and W. D. Lu, "Side-channel attack analysis on in-memory computing architectures," *IEEE Trans. Emerg. Topics Comput.*, vol. 12, no. 1, pp. 109–121, Jan. 2024.
[58] Z. Wang, Y. Wu, Y. Park, and W. D. Lu, "Safe, secure and trustworthy compute-in-memory accelerators," *Nature Electron.*, vol. 7, no. 12, pp. 1086–1097, Dec. 2024.
[59] B. Sapui and M. B. Tahoori, "Power side-channel analysis and mitigation for neural network accelerators based on memristive crossbars," in *Proc. 29th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2024, pp. 612–617.
[60] F. J. Mir, A. Aljuffri, S. Hamdioui, and M. Taouil, "Extracting weights of CIM-based neural networks through power analysis of adder-trees," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2024, pp. 1–4.
[61] M. Heidary and B. K. Joardar, "Hardware attacks on ReRAM-based AI accelerators," in *Proc. IEEE 17th Dallas Circuits Syst. Conf. (DCAS)*, Apr. 2024, pp. 1–4.
[62] B. K. Joardar and K. Chakrabarty, "Attacking ReRAM-based architectures using repeated writes," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2023, pp. 1–6.
[63] T. P. Xiao et al., "On the accuracy of analog neural network inference accelerators [Feature]," *IEEE Circuits Syst. Mag.*, vol. 22, no. 4, pp. 26–48, 4th Quart., 2022.
[64] S.-H. Chang, R.-H. Yen, and C.-N. Liu, "Error-aware training for in-RRAM computing design considering non-ideal effects in RRAM crossbar array and peripheral circuits," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 21, no. 2, pp. 1–22, Apr. 2025.

Ashish Reddy Bommana (Graduate Student Member, IEEE) received the B.Tech. degree from Indian Institute of Technology (IIT) Bhubaneswar, India, in 2021, and the M.S. degree from Arizona State University, Tempe, AZ, USA, in 2024, where he is currently pursuing the Ph.D. degree. He has published work in prestigious venues, including IEEE ITC, IEEE DATE, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, and *ACM Transactions on Design Automation of Electronic Systems*. His research interests include the design of compute-in-memory accelerators, AI hardware accelerators, and fault-tolerant deep learning systems. He has served as a reviewer for IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.

Rajendra Bishnoi received the Ph.D. degree in computer science from Karlsruhe Institute of Technology (KIT), Germany, in 2017. He was a Research Leader with the MRAM Group, for more than two years. From 2006 to 2012, he was a Design Engineer with Freescale (now NXP), where he was a part of the Technical Solution Group in memory and SoC flow. He is currently an Assistant Professor with the Computer Engineering Group, Delft University of Technology (TU-Delft). His current research interests include neuromorphic computing, computation-in-memory, and emerging technologies. He was a recipient of the EDAA Outstanding Dissertation Award in 2017. He has served on the TPC for several prominent conferences, including DAC 2006, ICCAD 2005, DAC 2005, DATE 2005, DATE 2004, ICCAD 2004, ETS 2004, DATE 2003, and DATE 2002.

Naghmeh Karimi (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer engineering from the University of Tehran, Iran, in 2002 and 2010, respectively. She was a Visiting Researcher with Yale University, New Haven, CT, USA, from 2007 to 2009; a Post-Doctoral Researcher with Duke University, Durham, NC, USA, from 2011 to 2012; and a Visiting Assistant Professor with New York University, New York, NY, USA, from 2012 to 2014, and Rutgers University from 2014 to 2016. In 2017, she joined UMBC, where she leads the Secure, Reliable, and Trusted Systems (SECRETS) Research Laboratory. She is currently an Associate Professor with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County (UMBC). She has published five book chapters and more than 110 papers in refereed conference proceedings and journal articles. Her current research interests include hardware security and design-for-trust, fault tolerance and design-for-reliability, VLSI testing and design-for-testability, as well as security and reliability of machine learning accelerators. She was a recipient of the National Science Foundation CAREER Award in 2020. She serves as an Associate Editor for the *Journal of Electronic Testing: Theory and Applications* (Springer) and IEEE DESIGN AND TEST journal. She has been the corresponding Guest Editor of *Journal on Emerging and Selected Topics in Circuits and Systems*, Special Issue in Hardware Security in Emerging Technologies in 2021.

Farshad Firouzi is currently a Faculty Member with Arizona State University and a Technology Entrepreneur. His professional experience includes research roles at world-leading technology institutions, such as imec. He serves as a PI/Co-PI on research projects funded by agencies, including the National Science Foundation (NSF) and the Semiconductor Research Corporation (SRC), and has successfully led proposals with a cumulative funding portfolio exceeding 10 million. He has authored more than 100 publications and edited/authored three books on artificial intelligence and the Internet of Things (IoT). He actively contributes to the research community as an Editorial Board Member for several leading journals, including the IEEE INTERNET OF THINGS JOURNAL, and through service on the program committees of premier conferences, such as COINS, DATE, DAC, ICCAD, HOST, and VTS.

Krishnendu Chakrabarty (Fellow, IEEE) is currently a Fulton Professor of Microelectronics with the School of Electrical, Computer and Energy Engineering, Arizona State University (ASU), and the Chief Technology Officer with the Department of War Microelectronics Commons Southwest Advanced Prototyping Hub. He is also the Director of the ASU Center for Semiconductor Microelectronics. Before joining ASU, he was the John Cocke Distinguished Professor and the Department Chair of Electrical and Computer Engineering with Duke University. His current research interests include design-for-test of 3-D integrated circuits, advanced packaging, and heterogeneous integration; AI hardware accelerators; hardware security; microfluidic biochips; and AI for healthcare. He is a fellow of the National Academy of Inventors, ACM, and AAAS. He was a recipient of many awards, including the Humboldt Research Award from the Alexander von Humboldt Foundation, Germany; the IEEE Computer Society Technical Achievement Award; the IEEE Circuits and Systems (CAS) Society Charles A. Desoer Technical Achievement Award; the IEEE CAS Society Vitold Belevitch Award; the Semiconductor Research Corporation (SRC) Technical Excellence Award; the SRC Aristotle Award; and the SRC Innovation Award. He has served as the Editor-in-Chief for IEEE DESIGN AND TEST, IEEE TRANSACTIONS ON VLSI SYSTEMS, and *ACM Journal on Emerging Technologies in Computing Systems*.