Extracting Weights of CIM-Based Neural Networks Through Power Analysis of Adder-Trees

Mir, Fouwad Jamil; Aljuffri, Abdullah; Hamdioui, Said; Taouil, Mottaqiallah

DOI
10.1109/ets61313.2024.10567473

Publication date
2024

Document Version
Final published version

Published in
2024 IEEE European Test Symposium (ETS)

Extracting Weights of CIM-Based Neural Networks Through Power Analysis of Adder-Trees

Fouwad Jamil Mir*[§], Abdullah Aljuffri*, Said Hamdioui*[§], Mottaqiallah Taouil*[§]
*TU Delft, Delft, The Netherlands [§]CognitiveIC, Delft, The Netherlands
Email: {F.J.Mir, A.A.M.Aljuffri, S.Hamdioui, M.Taouil}@tudelft.nl

2024 IEEE European Test Symposium (ETS) | 979-8-3503-4932-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ETS61313.2024.10567473

***Abstract*—Computation-in-Memory (CIM) architectures present a promising solution for efficient implementation of Neural Networks. Particularly, SRAM-based digital CIM architectures are optimal candidates to realize them. Recent studies have revealed potential weaknesses in these architectures, particularly against power attacks. This study introduces a novel attack method enabling weight extraction through the analysis of the adder tree component within the architecture. In our attack, the k-means clustering technique is employed to identify the hamming weights of the CIM weights. Subsequently, we correlate traces belonging to known weights with traces belonging to Hamming groups with unknown weights in order to identify their weight values. As a case study, the attack was applied on SRAM CIM implementation based on 40nm TSMC technology. The results indicate that the weights stored in the CIM crossbar can be retrieved with 100% accuracy purely by analyzing the power consumption.**

***Index Terms*—side-channel attack, CIM, Machine Learning, SRAM, Adder-tree**

I. Introduction

The traditional Von Neumann architecture, with its inherent limitations, poses a significant bottleneck for data-intensive applications such as Machine Learning (ML) and Artificial intelligence (AI) [1]. In order to address these limitations and meet the growing demand for ML and AI applications, novel architectures are being developed aimed at enhancing throughput and energy efficiency [2]. One such innovative architecture is Computation-in-Memory (CIM) that integrates storage and computing within the memory, thereby reducing time and energy spent on data fetching [3]. However, their focus on efficiency makes them vulnerable to attacks, such as side channel attacks, which can compromise valuable intellectual property (IP) of CIM-based Neural Networks (NN). Therefore, considering security threats during the design phase is crucial to ensure robust protection against such attacks.

Several studies in the literature have focused on side channel attacks aimed at extracting both micro-parameters (such as weights and biases) and macro-parameters (such as architecture, layer count, and activation functions) from NN models implemented on conventional microcontrollers and FPGAs [4, 5]. Furthermore, there has been a significant effort to exploit side-channel vulnerabilities in CIM based architectures, aiming to extract critical information including NN weights [6], NN taxonomy [7], input data [8], and secret keys [9]. For example, the attack presented in [7] reverse engineered a ReRAM-based NN model architecture using power side-channel attacks. On the other hand, Read et al. [6] presented an attack with the aim to steal the weights stored in emerging nonvolatile memory (eNVM) cells with 99% accuracy employing semi-invasive photonic emission analysis technique. Although various side-channel attacks on NN implementations within analog CIM devices have been studied in literature, research on the vulnerability assessment of digital CIM devices against such attacks is lacking.

This work presents a successful side-channel attack on a small-scale but representative digital CIM [10] comprising of a memory array, an adder tree and an accumulator register. The goal of this work is to explore a leakage model that allows us to use power side channel to extract the stored weights. It is important to mention that all the analysis are performed in the pre-silicon phase to explore zero-day vulnerabilities and mitigate them in actual implementations. To the best of our knowledge, this is the first attack carried out on digital CIM design. The contributions of this study are summarized as follow:

- Proposal of a novel approach to extract weights from digital CIM-based neural networks.
- Implementation of a small-scale but representative digital CIM macro [10] using 40nm CMOS technology as a case study.
- Validation of the proposed attack using gate-level implementation.

The rest of this paper is structured as follows. Section II provides a background on Digital SRAM CIM, power side-channel attacks and k-means clustering. Section III presents the methodology. Section IV presents the results. Section V discusses the limitations and concludes this work.

II. Background

This section provides a brief background on SRAM-based CIM, power side channel attacks, and k-means clustering.

A. SRAM CIM

Limited area and power footprint proposed by CIM makes them an ideal candidate for Deep Neural Networks (DNN) accelerators, especially in edge devices [11]. While eNVM-based CIM architectures face challenges

This work is funded by EU's Horizon Europe research and innovation programme under grant agreement No. 101070374.

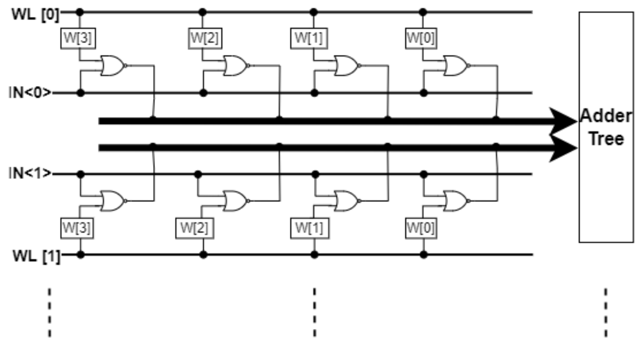


Fig. 1: Schematic of a Digital CIM [10]

like endurance and reliability which limits their use to inference-only accelerators, SRAM-based CIMs support both on-chip training and inference [6, 12]. SRAM-based CIMs can be realized in either analog or digital domain. Analog CIMs perform the MAC operation by multiplying weights stored in SRAM/ReRAM cells with inputs provided through word lines (WL), resulting in a current through bitlines (BLs) [13]. Digital CIM devices outperform their analog counterparts due to a mature CMOS technology and efficient realization [12]. The operands in digital CIM for vector matrix multiplication (VMM) are arranged in such a way that the input is provided as a vector using WLs, and the matrix is stored in crossbar within the SRAM cells. Bit-multiplication is performed using modified SRAM cells with additional transistors for logical operation. The result is sent to an adder tree using the BLs to compute the sum. A typical digital CIM *macro* is provided in [10]. It consists of an SRAM cell array, a dedicated adder tree for each of the 64 sub-CIM unit and an accumulation register to store partial sums. The internal structure of this sub-CIM unit is shown in Figure 1. It consists of a column containing 256x4 weight cells where the 256 inputs are simultaneously fed to the crossbar using WLs. The bit-wise multiplication, realized by a NOR operation (which in essence is an AND operation with inverted inputs), is performed inside the crossbar and the result is provided to the adder tree for sum computation.

B. Power Side Channel Attacks

Side channel attacks are hardware attacks that exploit unintended leaks (e.g power consumption, electromagnetic emissions, or timing information) to extract secret information from a system [4, 14]. Power side channel attacks (PSCA) extract information through the analysis of variations in power consumption during execution, such as identifying the switching power of registers [15].

There are various abstraction levels of PSCA. Simple power analysis (SPA) relies on the visual characteristics of the power trace, hence is the most basic type of attack. Correlation power analysis (CPA) relies on statistical methods to deduce a pattern between secret data and power consumption [4, 14, 15]. Advanced AI techniques like deep learning [16] are also employed for sophisticated side channel attacks. This work demonstrates that the digital SRAM CIMs are vulnerable to power attacks, utilizing ML approach such as k-means [17] to reveal the secret weights stored in SRAM cells.

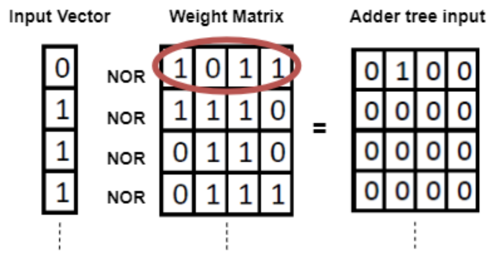


Fig. 2: Single Row Activation

C. K-means Clustering Algorithm

Clustering is a technique used to group data with similar attributes. The k-means algorithm [18] is widely used and highly recognized for its simplicity and efficiency. It partitions datasets into k clusters by iteratively reallocating each data point to the centroid with the lowest distance. The centroids are then updated by averaging the data points within each cluster. This iterative process continues until centroids stabilize or a set number of iterations is reached. Particularly suitable for our study, k-means identifies power characteristics based on their Hamming weight (HW), often revealing distinct and recognizable patterns.

III. Side-channel Attack Framework

This section describes the attack framework. It starts by motivating the attack followed by describing each framework step in detail.

A. Motivation

Training neural networks is a resource-intensive process, requiring significant time and financial investment, as discussed in Huang et al. [9]. The pre-trained weights of these networks, being intellectual property, hold considerable value. For instance, in scenarios such as commercially available autonomous vehicles, adversaries can exploit reverse engineering attacks on NN accelerators to steal weights for producing counterfeit devices or selling them to competitors. Therefore, it is imperative to identify and mitigate side-channel vulnerabilities in the pre-silicon phase before mass production.

In the design under attack shown in Figure 1, it is observed that bit-serial multiplication is performed as first step of execution. By manipulating the input, it is possible to confine the switching activity to a single 4-bit weight. As shown in Figure 2, the weight in the first row (circled in red) serves as our target weight. The NOR operation between input 0 and weight ‘1011’ in first row results in ‘0100’, while similar operations in the remaining rows where a 1 is applied as input results in all zeros. This allows us to compute the targeted HW by performing power analysis and ultimately derive the weight values. This process is repeated by setting the input to zero for the next rows, one at a time.

B. Threat Model

We assume that the target device operates in a non-trusted environment, allowing the adversary to perform power measurements. Additionally, the adversary has

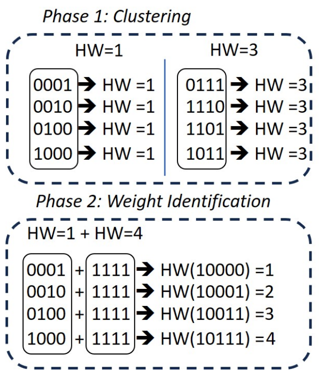

Phase 1: Clustering
HW=1
HW=3
0001 → HW =1
0010 → HW =1
0100 → HW =1
1000 → HW =1
0111 → HW =3
1110 → HW =3
1101 → HW =3
1011 → HW =3
Phase 2: Weight Identification
HW=1 + HW=4
0001 + 1111 → HW(10000) =1
0010 + 1111 → HW(10001) =2
0100 + 1111 → HW(10011) =3
1000 + 1111 → HW(10111) =4

Fig. 3: Illustration of the Attack Phases

physical and logical access to the device's data inputs. Furthermore, we assume the device is solely dedicated for inference operations with fixed weights. Although we consider that the attacker is acquainted with the accelerator's functionality, we assume that the attacker has no knowledge of the implemented NN.

C. Attack Methodology

Our methodology revolves around manipulating input values to incorporate or exclude specific 4-bit weights in the addition process by providing ones or zeros as input, enabling selective selection of weights in the SRAM. Our attack strategy, which is depicted in Figure 3, is divided into two phases. In Phase 1, weights are categorized based on their HW using clustering, as weights with identical Hamming values exhibit similar switching activities and hence power consumption. By varying input values, selective selection of weights and employing k-means clustering, we accurately determine the HW of each weight. The result of the clustering algorithm are 5 clusters identifying traces contain HWs of 0 to 4. Phase 2 utilizes weights with HW 4 (uniquely identified as value 15) to determine the remaining unknown weight values. This is achieved by activating and adding the known weights with unknown weights of same HW category. An example is shown in the bottom part of Figure 3 where weights with HW=1 can be identified when such weights are added with 15, as each result has a unique HW. This iterative process is optimized by employing exhaustive search to minimize additions, allowing deduction of all unknown weight values.

IV. Experimental Results

This section presents the experimental setup and results.

A. Setup

To validate the security of the SRAM-based CIM design against power side channel attacks, a combination of tools was employed: Questasim [19] for simulation, Cadence Genus [20] for synthesis, and Synopsys Spyglass [21] for generating power traces. The adder tree has been synthesized to a gate-level netlist using TSMC 40 nm technology. In this paper, we focused on 4-bit weight values, but the methodology is generally applicable. The attacks are carried out in Python using various supporting libraries such as scikit-learn [22].

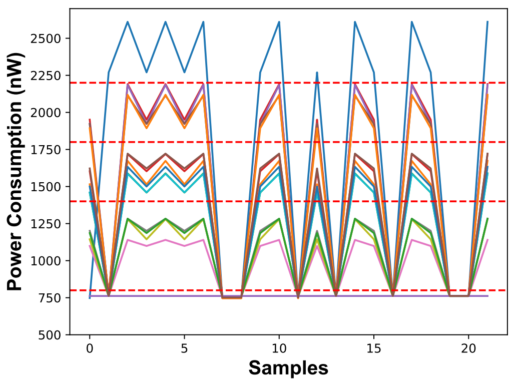


Fig. 4: First Phase: Clustering Results

B. Phase 1: Clustering

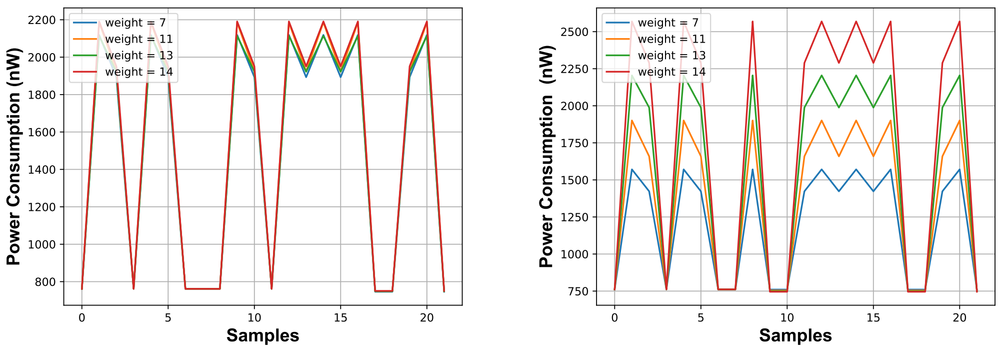
During the initial phase of the attack, we analyzed the power trace of each individual weight by activating only one of them at a time. Results for selected weights are shown in Figure 4, indicating a clear relationship between weight HW and power consumption patterns during adder tree operation. Power traces exhibited distinct clustering by the k-means algorithm, highlighting significant differences between each HW category.

C. Phase 2: Weight Identification

The next step is to determine the value of weights after determining the HW of each weight in phase 1. We analyze each HW set ranging from 0 to 4 obtained from clustering and start with traces in the sets with extreme HWs (0 and 4), whose corresponding values are known (0 and 15, respectively). Next, we analyze traces from the weights in the HW 1 set. To precisely quantify power patterns associated with these values, we activate two weights simultaneously: one unknown weight with HW 1 and a known weight with a value of 15. Subsequently, we move on to other HW values to acquire unique identifier. For example, Figure 5 illustrates the power consumption for the adder tree using unknown weights with HW 3 (values 7, 11, 13, and 14), with and without the activation of a known weight of value 1. This reveals distinct power patterns and HWs for each value (HW=1 for 7+1, HW=2 for 11+1, HW=3 for 13+1, and HW=4 for 14+1). Note that, despite the measurements being taken in a controlled and noise-free environment, the distinctive patterns among the various HW are quite evident. Therefore, this level of clarity suggests that the same traces would still be vulnerable to attacks even in the presence of noise.

V. Discussion & Conclusion

This study introduced a novel attack method on adder tree based digital SRAM CIMs. The first phase of the attack applies k-means clustering on power traces to identify the HWs of the weights, while second phase identifies the unknown weights by correlating them with known weight values. The attack achieves 100% extraction of NN weight values. Based on our results, we conclude the following:



(a) Single Weight Activation with HW=3

(b) Result of Adding a Weight with HW=3 and the Value 1

Fig. 5: Second Phase: Hamming Weight Three Results

Comparison with State-of-the-Art: Our study considered digital SRAM-based CIM as the target architecture, in contrast to other studies that focuses on ReRAM and analog SRAM. Wang et al. [7] conducted a non-profiled attack to extract the architecture information, while Ensan et al. [23] employed a template-based side-channel analysis (SCA) method to deduce the functionality of the system. Our study is unique as it demonstrates a novel non-profile attack on digital CIM that has not been explored previously.

Countermeasures: As our attack targets the HWs produced as partial sum, a practical solution is to mask this information such that the adversary is unable to distinguish between the HWs of dot products. For example, the technique proposed in [24] utilized masking as a countermeasure to thwart side-channel attacks on DNN accelerators. Masking breaks the input-result relationship by splitting weights into two columns through the addition and subtraction of a pseudo-random number, ensuring correctness of the final result.

SMT Solver: In the second phase, we combined known and unknown values with known HWs to identify the unknowns, aiming to find a suitable sum for each pair that yields a distinct HW. An SMT solver can enhance the efficiency of this search procedure, especially for larger weight sets. This integration optimizes the process, enhancing efficiency in handling complex scenarios and datasets.

REFERENCES

[1] X. Zou *et al.*, "Breaking the von neumann bottleneck: architecture-level processing-in-memory technology," in *Sci. China Inf. Sci.*, 2021.

[2] S. Hamdioui *et al.*, "Memrisor Based Computation-in-Memory Architecture for Data-Intensive Applications," in *DATE*, 2015, pp. 1718–1725.

[3] W. Chen *et al.*, "Resistive-RAM-Based In-Memory Computing for Neural Network: A Review," in *Electronics*, vol. 11, 2022, p. 3667.

[4] M.M. Real *et al.*, "Physical side-channel attacks on embedded neural networks: A survey," in *Applied Sciences (Switzerland)*, vol. 11, 2021.

[5] W. Liu *et al.*, "Two Sides of the Same Coin: Boons and Banes of Machine Learning in Hardware Security," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, 2021, pp. 228–251.

[6] J. Read *et al.*, "A Method for Reverse Engineering Neural Network Parameters from Compute-in-Memory Accelerators," in *IEEE ISVLSI*, 2022, pp. 302–307.

[7] Z. Wang *et al.*, "Side-Channel Attack Analysis on In-Memory Computing Architectures," in *IEEE TETC*, 9 2023, pp. 1–13.

[8] Z. Wang *et al.*, "PowerGAN: A Machine Learning Approach for Power Side-Channel Attack on Compute-in-Memory Accelerators," in *Advanced Intelligent Systems*, 9 2023.

[9] S. Huang *et al.*, "XOR-CIM: Compute-In-Memory SRAM Architecture with Embedded XOR Encryption," in *ICCAD*, 2020.

[10] Y. Chih *et al.*, "An 89tops/w and 16.3tops/mm^2 all-digital sram-based full-precision compute-in memory macro in 22nm for machine-learning edge applications," in *IEEE ISSCC*, 2021, pp. 252–254.

[11] X. Si *et al.*, "15.5 A 28nm 64Kb 6T SRAM Computing-in-Memory Macro with 8b MAC Operation for AI Edge Chips," in *IEEE ISSCC*, 2020, pp. 246–248.

[12] D. Li *et al.*, "All-Digital Computing-in-Memory Macro Supporting FP64-Based Fused Multiply-Add Operation," in *Applied Sciences (Switzerland)*, vol. 13, 2023.

[13] G.W. Burr *et al.*, "Ohm's Law + Kirchhoff's Current Law = Better AI: Neural-Network Processing Done in Memory with Analog Circuits will Save Energy," in *IEEE Spectrum*, vol. 58, 2021, pp. 44–49.

[14] A. Aljuffri *et al.*, "S-NET: A confusion based countermeasure against power attacks for SBOX," in *20th International Conference, SAMOS*, A. Orailoglu *et al.*, Eds., vol. 12471, 2020, pp. 295–307.

[15] M. Randolph *et al.*, "Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman," in *Cryptography*, vol. 4, 2020.

[16] A. Aljuffri *et al.*, "Impact of data pre-processing techniques on deep learning based power attacks," in *16th DTIS*, 2021, pp. 1–6.

[17] I. Kabin *et al.*, "Horizontal attacks using k-means: Comparison with traditional analysis methods," in *10th NTMS*, 2019, pp. 1–7.

[18] S. Lloyd, "Least squares quantization in PCM," in *IEEE Transactions on Information Theory*, vol. 28, 1982, pp. 129–137.

[19] Siemens, "Questa Advanced Simulator," https://eda.sw.siemens.com/en-US/ic/questa/simulation/advanced-simulator/, accessed:2023-09-08.

[20] Cadence, "Cadence Genus Sythensis Solution," https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html, accessed:2023-09-08.

[21] Synopsys, "Synopsys SpyGlass Power," https://www.synopsys.com/verification/static-and-formal-verification/spyglass/spyglass-power.html, accessed:2023-09-08.

[22] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] S.S. Ensan *et al.*, "SCARE: Side Channel Attack on In-Memory Computing for Reverse Engineering," in *IEEE VLSI*, vol. 29, 2021, pp. 2040–2051.

[24] A. Dubey *et al.*, "MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection," in *IEEE HOST*, 2020, pp. 197–208.