

Make it Darker: A Gray Code Popcounter to Protect BNN CIM Against Power Attacks

Fouwad Jamil Mir Asmae El Arrassi Abdullah Aljuffri Said Hamdioui Mottaqiallah Taouil
TU Delft / CognitiveIC *TU Delft* *TU Delft* *TU Delft / CognitiveIC* *TU Delft / CognitiveIC*
 Delft, The Netherlands Delft, The Netherlands Delft, The Netherlands Delft, The Netherlands Delft, The Netherlands
 f.j.mir@tudelft.nl a.elarrassi@tudelft.nl a.a.m.aljuffri@tudelft.nl s.hamdioui@tudelft.nl m.taouil@tudelft.nl

Abstract—Binary Neural Networks (BNNs) have obtained a strong foothold in the field of machine learning at the edge due to their minimal hardware requirements. However, their energy and performance efficiency remain hindered by frequent data transfer between memory and processors. Computation-in-memory (CIM) architectures address this problem by embedding processing units within the memory. Unfortunately, current implementations of CIM are susceptible to IP piracy attacks through side channels. This paper presents a novel secure periphery scheme for NN accelerators with sequential accumulation that conceals IP information by obscuring the power consumption of the counter responsible for the leakage. This is achieved by combining two innovative techniques: operand schedule randomization and an always-count Gray code counter. The results demonstrate that the proposed design effectively resists power side channel attacks (SCAs). Moreover, Signal-to-Noise Ratio (SNR) and Test Vector Leakage Assessment (TVLA) show safe leakage levels. Compared to the state-of-the-art, our countermeasure reduces area and power overheads by up to $12.7\times$ and $13.3\times$, achieving only 37% area and 51.2% power overhead with the added protection logic. Notably, this enhanced security comes with zero latency overhead, maintaining the performance of the baseline design.

Index Terms—Side Channel Attacks, CIM, Binary Neural Networks, Correlational Power Analysis, Countermeasures

I. INTRODUCTION

Deep Neural Networks (DNNs) have revolutionized machine learning (ML), achieving remarkable advancements across diverse domains such as computer vision [1] and natural language processing [2]. Cloud platforms utilize power-hungry GPUs to process large DNN models, which is impractical for edge computing [3]. BNNs enable the deployment of complex models on resource-constrained edge devices by utilizing single-bit weights and activations, reducing model size and computational requirements [4]. However, frequent memory-processor data movement in traditional von Neumann architectures limits efficiency, necessitating hardware innovation [5]. CIM architectures address this by co-locating processing units within the memory, significantly reducing energy consumption [6]. Despite these benefits, they face security challenges, particularly reverse-engineering attacks targeting proprietary weights and topology. SCAs, such as Correlational Power Analysis (CPA), exploit power consumption to extract such secret information. Digital BNN-CIM designs are more susceptible to side-channel

leakage due to the distinct switching patterns of binary operations, hence require mitigation strategies.

Previous research has explored SCA vulnerabilities in ReRAM-based analog CIM architectures [7–9]. However, these studies do not fully address the unique security challenges posed by digital CIM-based BNNs. Limited work has addressed BNNs implemented on FPGAs [10, 11]. They focus on masking-based countermeasures, which provide dependable security, but they involve computationally expensive PRNGs, hindering their applications in digital BNN-CIM implementations, where strict area/power constraints make high-overhead techniques infeasible. The discrete nature of binary operations in digital CIM highlights side-channel leakage through distinct power consumption patterns. Masking, for instance, incurs significant area overhead (up to $5\times$) and reduces throughput [11]. This underscores the need for a thorough security analysis and the development of tailored defenses with minimal overhead.

In this study, we analyze the security of a state-of-the-art digital BNN CIM architecture against SCAs. Although proposed but not implemented by Dubey et al. in [12], this is, to the best of our knowledge, the first temporal shuffling scheme with simultaneous power equalization in digital hardware. Our novel countermeasure obscures power consumption by obfuscating the accumulation order and equalizes switching activity for zero and one inputs, preventing adversaries from inferring the count. As a result, the proposed countermeasure offers strong resilience against CPA with minimal area and power overhead. The key contributions of this work are:

- Conducting a comprehensive security analysis of a sequential BNN-CIM accelerator using a CPA model specifically designed for serialized accumulation;
- Proposing a lightweight, hardware-friendly operand schedule randomization mechanism complemented with a novel power equalization scheme, significantly enhancing the security of BNN accelerator;
- Implementing the proposed countermeasure using 40nm TSMC node technology;
- Validating the countermeasure’s effectiveness through CPA [13], Welsh t-tests [14], and information-theoretic statistics for shuffling-based countermeasures [15].

The subsequent sections are structured as follows. Section II provides background on CIM and security metrics. Section III outlines the leakage, threat, and attack model. Section

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

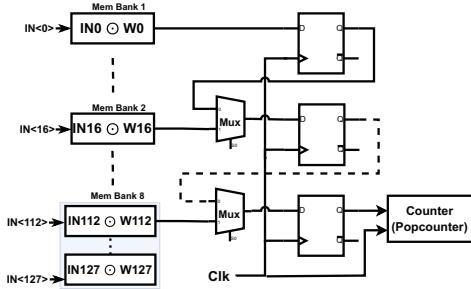


Fig. 1: Periphery Structure for Popcount Implementation [19]

IV explains the proposed mitigation scheme, followed by the results in Section V. Section VI presents the discussion and limitations, and Section VII concludes the work.

II. BACKGROUND

This section introduces BNN CIMs and SCAs.

A. Binary Neural Network based Digital CIM Macro

CIM integrates data storage and processing within memory units, reducing data movement and improving energy efficiency [6]. In digital CIM, SRAM cells perform vector-matrix multiplications (VMM) using embedded logic [16]. Inputs are applied to word lines (WLs), and weights are stored in columns, enabling in-memory multiplication and accumulation (MAC). Architectures are typically classified as sequential or parallel based on the accumulation strategy. Sequential designs reduce area and power but incur latency, while parallel designs offer higher throughput at greater resource cost [17–19].

This work takes one of the state-of-the-art sequential BNN accelerator [19] to evaluate our countermeasures. It replaces adder trees with shift-register and counter-based accumulation for better efficiency. The macro is based on a 10-T SRAM cell design capable of performing logical operations within the cell (XNOR in our case, representing binary multiplication). Each column has a read bit line (RBL) that outputs XNOR results, stored in flip-flops (FFs) and read via a scan chain into a popcounter. Input vectors are applied sequentially, activating one row per bank in parallel.

BNNs, with weights and inputs reduced to single-bit, enable efficient inference on resource-constrained devices [20]. Multiplications are replaced with XNOR operations, followed by a popcount, expressed as $MAC(\mathbf{IN}, \mathbf{W}) = \text{popcount}(\mathbf{IN} \odot \mathbf{W})$, where -1 is encoded as 0 and +1 as 1. This quantization significantly reduces computational overhead while maintaining acceptable accuracy [1, 21]. The macro is optimized for BNN acceleration, offering energy-efficient, high-density processing suitable for edge AI applications.

B. Security Metrics for Weight Extraction and Protection

We discuss standard metrics and models to evaluate resistance against correlation-based and profiling SCAs.

Statistical Leakage Metrics. SNR [22] measures the ratio of data-dependent variance to noise, while TVLA [14] applies Welch’s t -test to detect statistically significant differences between fixed and random input sets.

CPA. CPA correlates measured power traces with hypothetical leakage values derived from guessed secrets [13]. The Pearson correlation coefficient is computed per time sample, and attack success depends on SNR and trace count T , with $T \propto 1/\text{SNR}^2$ under Gaussian assumptions [23].

Alignment Probability. Temporal randomization disrupts the fixed alignment assumption of CPA. If the probability of micro-operation executing at the same time index across traces is p , the correlation reduces, and the trace complexity (i.e., the number of required traces) scale as $T \propto 1/p^2$ [23].

Windowed Attacks. Sliding-window attacks mitigate global misalignment by scanning local windows. For N window slots of size w , the success probability per trace is $p_{\text{win}} = w/N$, and the trace complexity scales as $(N/w)^2$ [24].

Profiling Attacks (Template and Deep Learning). Profiling attacks build statistical or ML models of leakage. A standard metric for quantifying class separability is the Kullback–Leibler (KL) divergence, defined as

$$D_{\text{KL}}(P \parallel Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \quad (1)$$

Here, P and Q correspond to the leakage distributions of different hypotheses (e.g., two key classes). Larger divergence implies greater distinguishability, whereas smaller divergence indicates overlap and higher profiling complexity. Operand shuffling transforms aligned leakage into a mixture of shifted components, reducing divergence between classes by approximately $1/N$ under the sparse-slot model [15]. Since sample complexity scales inversely with squared divergence, the required profiling traces increase by N^2 [22, 25, 26].

III. LEAKAGE, THREAT AND ATTACK MODEL

This section explains the leakage, threat, and attack model.

A. Leakage Model

After training, the topology and operations of NN accelerators are typically fixed. CIM architectures are inherently secure against memory-read attacks as multiplication is performed within memory cells rather than by fetching operands into separate multipliers. However, in sequential BNN-CIMs, CPA can target operations at the bit-serial accumulation stage. This vulnerability arises because such operations follow predictable data flow patterns that may leak exploitable information. Specifically, the use of shift registers and counters for accumulation in place of parallel adder trees creates a repetitive and predictable switching activity. These sequential elements, particularly the popcounter, exhibit strong correlations between their internal state transitions and the processed data. Since the input vectors are typically controllable by an adversary (e.g., during inference), and the accumulation process is deterministic, the resulting power consumption becomes a reliable source of leakage. In our target architecture, the popcounter accumulates partial bit-multiplication results by counting the logical ‘1’s per column. Intermediate results are stored in FFs, one per memory bank, and are serially shifted into a counter over multiple cycles. Each shift cycle increments the counter based on the current bit value, producing power signatures that correlate with the HW or HD of the intermediate values. The

temporal power signature of the counter increments becomes the dominant leakage source due to the serialized accumulation, which we utilize to build our CPA leakage model.

B. Threat Model

The threat model is built upon the assumptions of a gray-box attack, where the attacker has physical access to the device and can measure its power consumption during inference. In our model, we assume that the adversary is aware of the hardware architecture (i.e., SRAM-based CIM) and functionality (i.e., BNN accelerator) of the target design. However, the attacker does not possess any knowledge regarding the specific implementation details of the NN (e.g., weights, topology). We consider a non-invasive attack scenario where the attacker measures the power consumption of the CIM macro but cannot directly access the memory cells.

C. Attack Model: Correlation Power Analysis (CPA)

We propose a novel CPA-based attack methodology specifically tailored for XNOR-popcount-based sequential BNN accelerators. Unlike conventional attacks, our approach targets temporal leakage patterns introduced by serialized accumulation in the periphery. The hypothetical leakage model is defined as: $Hyp(W, IN) = HW(C(W \odot IN))$, where W and IN are 4-bit values, \odot denotes bitwise XNOR, and $C(\cdot)$ represents the intermediate counter state. The HW of this state reflects the switching activity contributing to power consumption. The 128-bit input vector is segmented into 4-bit chunks. For each chunk, Hyp values are generated for all 16 possible W guesses, representing popcount values that influence counter switching.

Power traces are then statistically correlated with these hypothetical leakage values using the Pearson correlation coefficient. A 4-bit granularity is chosen to balance computational complexity and attack resolution, allowing distinct correlation peaks for correct weight guesses. For each target weight, the hypothesis yielding the highest correlation peak is selected. A statistically significant peak strongly indicates a correct guess, allowing extraction of the stored weights.

IV. PROPOSED SECURE BNN ACCELERATOR DESIGN

This section describes the proposed countermeasure.

A. Motivation

Sequential BNN accelerators exhibit deterministic bit-serial popcount behavior, creating temporal correlations in power traces exploitable via CPA. An adversary controlling input vectors can align traces across multiple executions and infer weights based on whether partial XNOR results evaluate to one or zero. To mitigate this, we introduce a two-tiered countermeasure: (1) operand shuffling to randomize accumulation order and disrupt trace alignment, and (2) leakage equalization to ensure a uniform power profile regardless of whether a zero or one is processed. These are implemented by incorporating the following units in the architectural periphery (see Figure 2): a **Scrambler Unit**, which dynamically reorders FF outputs in the scan chain, and an **Always-count Gray Code Counter**, which maintains constant HD transitions, even for zero-valued

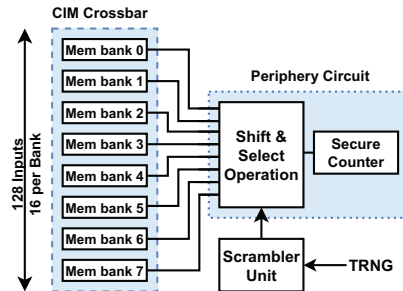


Fig. 2: Block Diagram of Side-Channel Secure Periphery

XNORs. We first elaborate on the theoretical strength of our proposed countermeasure, followed by the architectural implementation of the periphery.

B. Theoretical Strength

The proposed countermeasure increases the computational complexity of SCAs through both temporal randomization and information equalization. We analyze its effectiveness using two complementary models: trace complexity scaling and leakage observability.

Trace Complexity Scaling: Operand shuffling randomizes the execution cycle, making each popcount operation occur at a pseudorandom time offset across traces, breaking the fixed-alignment assumption of CPA. For 128 partial products randomized over 8 banks and 16 rows, the probability that the same pattern repeats at the same time in two traces is $p = (1/8)^{16} \approx 3.55 \times 10^{-15}$. This makes the effective SNR for correlation decrease by a factor of $1/p$, and the number of traces T required for a successful attack increases quadratically as $T \propto (1/p^2)$ [23]. For our parameters this pushes T well beyond any practical acquisition budget, effectively ruling out CPA and profiling attacks.

Leakage Equalization: To prevent windowed CPA or multivariate template attacks, which search for correlation within sliding trace windows [24], we introduce an Always-count Gray code counter. Unlike traditional binary counters, which exhibit variable HD and HW (e.g., 0111 to 1000 transitions), our design maintains a constant HD of 1 per transition. Additionally, it alternates between increments/decrements when the input is zero, ensuring uniform power profiles across all operations.

C. Scrambling Unit

The scrambling unit breaks the relationship between the popcount result and the execution order, protecting against attacks that exploit the sequential additions in the popcounter. It randomly selects one of the eight register values using a multiplexer and then passes the selected value to the secure counter. To generate the pseudorandom sequence, the unit incorporates a nonlinear feedback shift register (NFSR). Unlike a linear feedback shift register (LFSR) that cycles through a maximum of $n-1$ states, the NFSR traverses all n (eight) states, ensuring the selection of all values. Additionally, the NFSR seed is refreshed after every loop completion, minimizing the likelihood of pattern repetition and enhancing attack resistance.

The scrambling unit utilizes Rule 45 cellular automata (CA) block to generate a new seed every 8 clock cycles [27].

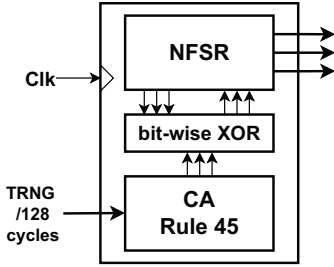


Fig. 3: Block Diagram and Internal Structure of Scrambler

Rule 45 is chosen among other CA rules because of its ability to generate chaotic, non-repeating patterns [27]. It spans through 2^m states for any m -bit value, ensuring uniformity and unpredictability of the sequence, and is less expensive to implement [27, 28]. It is defined by Equation 2 where s_i^t represents the state of the cell i at time t , and the function f encodes the Rule 45 logic. The rule is expressed in binary form as 00101101 (decimal 45), where each bit corresponds to the next state for a given combination of $(s_{i-1}^t, s_i^t, s_{i+1}^t)$.

$$s_i^{t+1} = f(s_{i-1}^t, s_i^t, s_{i+1}^t) \text{ where } s_i = s_{i-1} \oplus (s_i \vee (\neg s_{i+1})) \quad (2)$$

Before processing the next 8 partial products, the initial seed of NFSR is updated by XORing the output of the CA unit and the current NFSR value. This dependency-breaking mechanism produces a new seed as a quadratic non-linear function of both current and previous values, preventing its derivation solely as a time-dependent sequence and guarding against back-tracing. The combined CA–NFSR system has high algebraic and linear complexity, making its output statistically complex for modeling and hence cannot be captured by CPA. The security of the pseudo-random number generator (PRNG) is enhanced further by seeding the NFSR with a true random number after every 128 cycles (end of 1 MAC stage).

D. Always-count Gray Code Counter

A popcounter is inherently vulnerable to side channel leakage due to the observable HD between consecutive counted values. To address this, we implemented a Gray code counter, which maintains an HD of one between successive values, minimizing leakage during transitions. However, Gray coding alone is insufficient, as attackers can exploit other properties like the HW of the counter value, especially in correlation-based attacks. Additionally, the deterministic increment pattern of counter can reveal sensitive information. For example, increments occur only when the input to the popcounter is one, allowing an adversary to correlate power signatures with specific inputs.

To address these issues, we introduce an alternating always-count counter based on an increment/decrement mechanism. When the input is zero (XNOR=0), the counter alternates between incrementing and decrementing, introducing ambiguity in the trace. This alternation does not affect the final count, provided an even number of zeros are processed. In case of an odd number of zeros, a 1-bit flag register, tracking zero-count parity, is set and decrements the counter value by one after 128 inputs for a neuron. The working principle of the secure counter is illustrated with two examples in Table I.

TABLE I: Comparison of Regular and Secure Counter Behavior

Type	Clock Cycle	1	2	3	4	5	6	7	8
Regular	Bin. Seq.	0	1	0	1	0	0	1	0
	Value HW / HD	0 0/0	1 1/1	1 1/0	2 1/2	2 1/0	2 1/0	3 2/1	3 2/0
Secure	Bin. Seq.	1	1	0	1	1	0	1	1
	Value HW / HD	1 1/1	2 1/1	1 1/1	2 1/1	3 2/1	2 1/1	3 2/1	4 1/1
Regular	Bin. Seq.	1	2	2	3	4	4	5	6
	Value HW / HD	1 1/1	2 1/2	2 1/0	3 2/1	4 1/3	4 1/0	5 2/1	6 2/2
Secure	Bin. Seq.	1	2	3	4	5	4	5	6
	Value HW / HD	1 1/1	2 1/1	3 2/1	4 1/1	5 2/1	4 1/1	5 2/1	6 2/1

In the baseline design, each increment leads to observable changes in HW, HD, or both, directly exposing the counter value and the corresponding weight. In contrast, the secure counter maintains a constant HD across all transitions, while HW variations, whether they occur or not, result from either a 0 (due to alternating increment/decrement operation) or 1 input. This ambiguity significantly hinders attacker’s ability to correlate power traces with actual input data.

V. EXPERIMENTATION RESULTS

This section presents the experimental setup and results.

A. Setup

The proposed design is implemented in Cadence Virtuoso using TSMC 40 nm technology. Initial SPICE-level simulations performed in Cadence Spectre revealed obvious signs of side-channel leakage, even after introducing 20 dB transient noise as suggested in literature [29]. Consequently, we opted for hardware implementation to obtain realistic measurements. For this purpose, both the baseline and the secure version of the digital periphery were implemented on the ChipWhisperer CW305 development board, which features an Artix XC7A100T FPGA. All results reported in subsequent Sections are exclusively derived from power measurements captured on this platform. This setup serves three key purposes: (1) to access post-silicon leakage behavior and verify the presence of exploitable vulnerabilities, (2) to evaluate the effectiveness of the proposed countermeasures using realistic traces, and (3) to benchmark performance overheads in comparison with masked FPGA-based sequential BNN accelerators [11]. Experiments were conducted at an operating frequency of 25 MHz, with power traces sampled at 100 MS/s to meet the Nyquist criterion. A total of one million traces were collected for all the analyses that were carried out using Python.

B. PRNG Statistical Assessment

To assess the robustness of the scrambling unit, its PRNG sequence is evaluated for statistical properties such as uniformity and randomness. These metrics are critical to make sure that the execution order is free from predictable patterns, thus mitigating correlation-based attacks.

a) Uniformity: Uniformity is a vital trait for our PRNG that ensures each scrambling unit sequence is equally likely to occur. We validated this by generating 10,000 seeds (80,000 scrambling unit cycles) across three cases where the one input

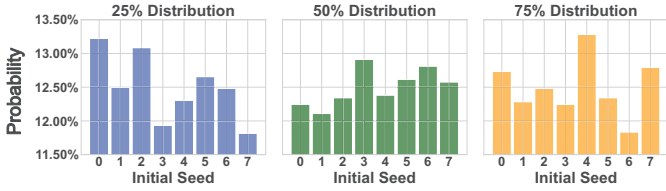


Fig. 4: Uniformity Evaluation of Seeds Generated for New NFSR Cycle

TABLE II: NIST Randomness Test Results for PRNG Output

Test Name	p-value	Result
Monobit Frequency	0.98	Pass
Block Frequency	0.55	Pass
Runs	0.24	Pass
Longest Run of 1s	0.74	Pass
Binary Matrix Rank	0.29	Pass
Linear Complexity	0.85	Pass
Serial	0.86	Pass
Approx. Entropy	0.08	Pass
Cumulative Sums	0.88	Pass

(XNOR=1) ratio is 25%, 50%, and 75%. Figure 4 shows the occurrence probability of the initial seeds. The Chi-square p-values are 0.68, 0.79, and 0.61, respectively. These results ($p > 0.05$) confirm uniformity in all cases, demonstrating robustness across varying XNOR distribution variations.

b) NIST random test suite: Randomness is a critical performance metric for PRNGs. We evaluated our PRNG sequence using the NIST test suite [30]. The test was applied to a sample size of 300×10^3 bits, split into 24-bit blocks. Each block consisted of all NFSR values during the 8 shifts, i.e., 12.5K scrambler rounds were evaluated, each generating $8 \times 3 = 24$ bits. Table II presents the results and corresponding p -values.

C. SNR Comparison Between Two Counters

SNR quantifies how distinguishable data-dependent variations (signal) are from random variations (noise) in power traces. Higher SNR indicates a stronger correlation between the observed leakage and the sensitive variables, facilitating recovery of secret data. As shown by Standaert et al. [22], reducing SNR increases leakage entropy, complicating effective leakage model building (e.g., in CPA) and lowering probability of successful attacks. To assess our countermeasure, we compare the SNR of the baseline and secure counter implementations, isolating them from periphery and control signals. As shown in Figure 5, the protected design achieves ~ 6 dB peak SNR reduction. The average SNR drops from 6.643dB to -1.761dB, significantly lowering exploitable leakage and improving resistance to CPA attacks.

D. Security Evaluation through TVLA

TVLA applies Welch's t-test to statistically detect data-dependent leakage in power traces [14]. A t-value exceeding the threshold of ± 4.5 indicates exploitable leakage. Following best practices for evaluating hiding countermeasures [31, 32], we adopted a semi-fixed vs. random input method and tailored it for NN accelerators. Two sets of traces were collected: one

TABLE III: Trace Misalignment Statistics

Metric	Ideal	Calculated
Entropy (bits)	3.000	2.943
Coverage (% of 8)	100.00%	89.34%
Collision	0.000	0.153
Auto Corr.	0.000	+0.003

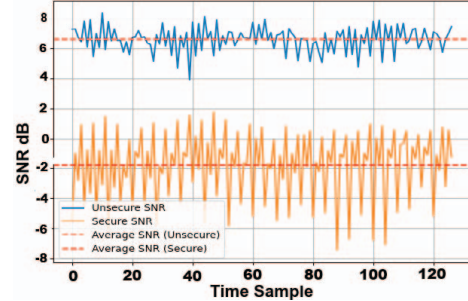


Fig. 5: SNR Comparison of Unsecured and Secure Counters

with semi-fixed inputs and the other with random inputs, while keeping the weights constant. Specifically, 4 consecutive input bits were varied while the rest were fixed, enabling precise localization of leakage samples for a specific bit subset, later used in Section V-E for CPA evaluation. Figure 6 presents the TVLA where the unsecured implementation shows all t-values exceeding the ± 4.5 threshold, indicating strong data-dependent leakages. The first 384 samples show lower t-values due to increased entropy (effect of varying vs fixed inputs). In contrast, the secure design consistently shows all t-values within the safe region of ± 4.5 , validating the effectiveness of our countermeasures.

E. Security Evaluation through CPA

A detailed description of the CPA-based attack method was provided in Section III-C. To evaluate the vulnerability of the baseline design and the effectiveness of the countermeasure, we perform CPA targeting intermediate computations during inference. The 128-bit input is split into 4-bit chunks, and the HW of XNOR results between known inputs and guessed weights is used as the leakage model.

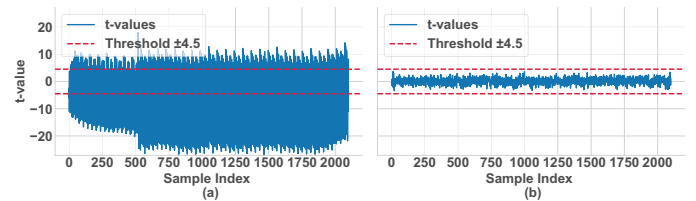


Fig. 6: Comparison of TVLA Results Without (a) and With (b) Countermeasure

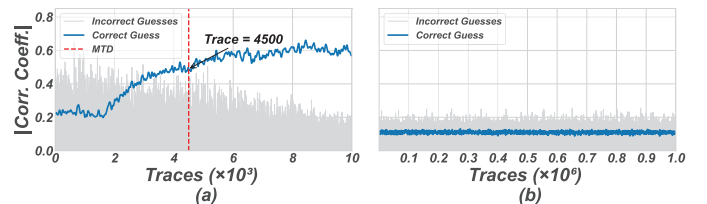


Fig. 7: CPA Analysis of Unsecure (a) and Secure (b) Counter

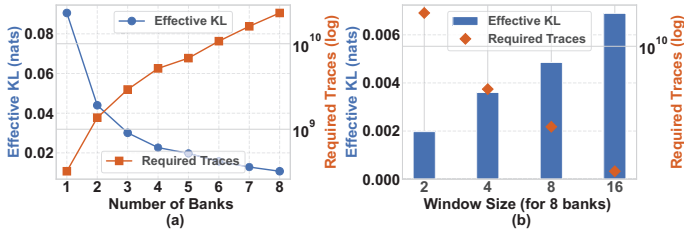


Fig. 8: Profiling Complexity Under Operand Shuffling: (a) KL divergence vs. banks, (b) Windowed attack cost.

We assessed the attack success by correlating hypothetical leakage with measured power traces across varying guessed weights. The results in Figure 7, reveal that the unprotected design has an MTD of 4,500 traces for the correct weights guess, demonstrating the design’s vulnerability. Conversely, the secure implementation remained resilient, with a consistently low r – values up to 1 million traces, highlighting the effectiveness of the implemented countermeasures.

F. Protection Against Profiling

Operand shuffling combined with the observed SNR drop (6.64 dB \rightarrow -1.76 dB) inflates the cost of profiling attacks by contracting inter-class divergence and misaligning the points of interest. Figure 8(a) reports the measured KL divergence (Eq. 1) between leakage classes under different shuffle configurations. We observe a clear decrease in divergence as the number of shuffled banks increases. Figure 8(b) further illustrates that even windowed attacks remain impractical. Although increasing the window size reduced the trace count, our measurements show that the associated computational effort grows rapidly, and beyond a moderate window length, the required capacity becomes infeasible. To validate trace misalignment, we collected traces with fixed inputs and weights, measuring repetition probability of the same micro-operation across 16 runs per bank. Table III reports that the entropy approaches the ideal 3 bits, coverage is near complete, collisions remain low, and autocorrelation is negligible. These results confirm that our countermeasure produces sufficiently unpredictable execution orders, compounding the SNR penalty into a profiling complexity that exceeds any realistic acquisition budget.

G. Accuracy Comparison

We evaluated the impact of our secure architecture on inference accuracy using the same setup as the baseline [19]. When tested on a BNN with LeNet-5 trained on MNIST, our design incurred only a 0.48% drop in classification accuracy, which is attributed to the gray counter. If a gray decoder is used, the error will be 0. However, we deem the 0.48% accuracy loss acceptable. Unlike prior work [33], our approach does not require retraining the network, avoiding additional time and resource overhead.

H. Overhead Assessment

Our approach is designed to hide power leakage with minimal overhead. To the best of our knowledge, no prior FPGA/ASIC implementation of shuffling-based countermeasures exists for BNN accelerators, making a direct comparison

TABLE IV: Overhead Comparison with Masked Design

Metric	Type	Baseline	Masked [11]	This work
		FPGA		
Area	LUTs	193	247%	33.6%
	FFs	26	657%	100%
Power (μ W)	—	198	472%	36.8%
Latency (cycles)	—	128	3%	0%
ASIC (40nm TSMC)				
Area (μ m ²)	—	498.1	470%	37%
Power (μ W)	—	61.7	683%	51.2%
Latency (cycles)	—	128	3%	0%

infeasible. Benchmarking against μ Controller-based software shuffling is impractical due to platform differences [24, 34]. A relevant comparison can only be performed with traditional masking approaches for sequential BNN accelerators. To ensure fair assessment, we implemented the masking scheme proposed in [11] in our design for overhead evaluation on ASIC/FPGA platforms. On ASIC, our secure design incurs 37% area and 51.2% energy overhead, versus 470% area and 683% power overhead for masking (see Table IV). On FPGA, our secure design adds 33.6% LUT and 36.8% power overhead, compared to 247% and 472% for the masked counterpart, respectively. Throughput remains unaffected, resulting in 0% overhead in latency. This balance of security and efficiency makes our countermeasure viable for resource-constrained environments.

VI. DISCUSSIONS

The proposed hiding countermeasure offers **practical** resistance by significantly increasing the trace complexity required for CPA and profiling [23]. The low overhead makes it highly practical for resource-constrained edge devices, providing a meaningful first line of defense. At the same time, it exhibits clear **scalability**. For instance, our randomizing technique can also protect higher-level data structures (e.g., MAC partial-product registers, input activation sequences) in any NN computation where MAC order does not affect correctness. Similarly, our secure counter is adaptable not only to other serial accumulation NN accelerators but also to components like program counter registers in side-channel secure microprocessors. Moreover, the integration of BNNs into complex SoCs introduces inherent noise and timing variability from parallel execution. These overlapping power signatures, combined with our spatial and temporal randomization, further disrupt exploitable leakage correlation under real-world conditions.

VII. CONCLUSION

This paper presented a novel lightweight countermeasure for the BNN CIM architecture, offering protection against CPA attacks with minimal area and energy overhead while maintaining the same throughput. We demonstrated that the unprotected BNN-CIM designs are susceptible to model theft attacks. In contrast, our design, combining randomized accumulation and balanced switching, significantly suppresses leakage across SNR, TVLA, and CPA tests. Information-theoretic profiling analysis further demonstrated that profiling costs grew logarithmically and exceeded feasible attack budgets. In summary, our countermeasure achieves strong side-channel resistance at low cost, making BNN CIM accelerators practical for edge AI.

REFERENCES

- [1] Y. LeCun *et al.*, “Deep learning,” *Nature*, vol. 521, pp. 436–444, 5 2015.
- [2] H. Li, “Deep learning for natural language processing: advantages and challenges,” *National Science Review*, pp. 24–26, 1 2018.
- [3] S. Huang *et al.*, “New Security Challenges on Machine Learning Inference Engine: Chip Cloning and Model Reverse Engineering.”
- [4] H. Qin *et al.*, “Binary neural networks: A survey,” *Pattern Recognition*, vol. 105, p. 107281, 9 2020.
- [5] S. Hamdioui *et al.*, “Memrisor Based Computation-in-Memory Architecture for Data-Intensive Applications,” in *DATE*, 2015.
- [6] W. Chen *et al.*, “Resistive-RAM-Based In-Memory Computing for Neural Network: A Review,” *Electronics*, vol. 11, p. 3667, 11 2022.
- [7] S.S. Ensan *et al.*, “SCARE: Side Channel Attack on In-Memory Computing for Reverse Engineering,” *IEEE VLSI Systems*, 2021.
- [8] J. Read *et al.*, “A Method for Reverse Engineering Neural Network Parameters from Compute-in-Memory Accelerators,” in *IEEE ISVLSI*, vol. 2022-July. IEEE, 7 2022, pp. 302–307.
- [9] Z. Wang *et al.*, “Side-Channel Attack Analysis on In-Memory Computing Architectures,” *IEEE TETC*, pp. 1–13, 9 2023.
- [10] A. Dubey *et al.*, “MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection,” in *IEEE HOST*, 2020.
- [11] A. Dubey *et al.*, “BoMaNet,” in *ICCAD*. ACM, 2020.
- [12] A. Dubey *et al.*, “Guarding Machine Learning Hardware Against Physical Side-channel Attacks,” *ACM JETC*, vol. 18, 4 2022.
- [13] E. Brier *et al.*, “Correlation Power Analysis with a Leakage Model,” *Lecture Notes in Computer Science*, vol. 3156, pp. 16–29, 2004.
- [14] G. Goodwill *et al.*, “A testing methodology for sidechannel resistance validation,” Tech. Rep.
- [15] J. Hermelink *et al.*, “Adapting Belief Propagation to Counter Shuffling of NTTs,” *Cryptology ePrint Archive*, 2022.
- [16] H. Kim *et al.*, “A 1-16b Precision Reconfigurable Digital In-Memory Computing Macro Featuring Column-MAC Architecture and Bit-Serial Computation,” in *IEEE ESSCIRC*, 2019.
- [17] A. Agrawal *et al.*, “IMPULSE: A 65-nm Digital Compute-in-Memory Macro with Fused Weights and Membrane Potential for Spike-Based Sequential Learning Tasks,” *IEEE L-SSC*, 2021.
- [18] H. Hu *et al.*, “Simulation of a Fully Digital Computing-in-Memory for Non-Volatile Memory for Artificial Intelligence Edge Applications,” *Micromachines* 2023, vol. 14, p. 1175, 5 2023.
- [19] A. El Arrassi *et al.*, “AFSRAM-CIM: Adder Free SRAM-Based Digital Computation-in-Memory for BNN,” in *VLSI-SoC*, 2024.
- [20] X. Sun *et al.*, “Fully parallel RRAM synaptic array for implementing binary neural network with (+1, 1) weights and (+1, 0) neurons,” in *IEEE ASP-DAC*, 2018.
- [21] M. Rastegari *et al.*, “XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks.” arXiv, 2016.
- [22] F.X. Standaert *et al.*, “Towards Security Limits in Side-Channel Attacks,” *Lecture Notes in Computer Science*, 2006.
- [23] S. Mangard *et al.*, *Power Analysis attacks: Revealing the secrets of smart cards*. Springer US, 2007.
- [24] M. Brosch *et al.*, “Counteract Side-Channel Analysis of Neural Networks by Shuffling,” *Proceedings of the DATE 2022*, 2022.
- [25] S. Chari *et al.*, “Template Attacks,” *CHES*, 2003.
- [26] H. Maghrebi, “Deep Learning based Side Channel Attacks in Practice,” *IACR Cryptol. ePrint Arch.*, 2019.
- [27] S. Wolfram, “Random sequence generation by cellular automata,” *Advances in Applied Mathematics*, vol. 7, pp. 123–169, 6 1986.
- [28] W.L. Dunn *et al.*, “Pseudorandom Number Generators,” in *Exploring Monte Carlo Methods*. Elsevier, 2023, pp. 55–110.
- [29] N. Gong *et al.*, “Signal and noise extraction from analog memory elements for neuromorphic computing,” *Nature Communications*.
- [30] A. Rukhin *et al.*, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” NIST, Tech. Rep., 2010.
- [31] U. Rioja *et al.*, “The Uncertainty of Side-channel Analysis: A Way to Leverage from Heuristics,” *ACM JETC*, 2021.
- [32] T. Schneider *et al.*, “Leakage assessment methodology,” in *CHES 2015*. Springer-Verlag, 2015.
- [33] M. Ashok *et al.*, “Digital In-Memory Compute for Machine Learning Applications With Input and Model Security,” *IEEE Journal of Solid-State Circuits*, pp. 1–13, 2025.
- [34] M.M. Real *et al.*, “Physical side-channel attacks on embedded neural networks: A survey,” *Applied Sciences (Switzerland)*, 2021.