POLARIS: Explainable Artificial Intelligence for Mitigating Power Side-Channel Leakage

Tanzim Mahfuz[1], Sudipta Paria[2], Tasneem Suha[1], Swarup Bhunia[2], and Prabuddha Chakraborty[1]

[1]Department of Electrical & Computer Engineering, University of Maine, Orono, ME, USA
[2]Department of Electrical & Computer Engineering, University of Florida, Gainesville, FL, USA

***Abstract*—Microelectronic systems are widely used in many sensitive applications (e.g., manufacturing, energy, defense). These systems increasingly handle sensitive data (e.g., encryption key) and are vulnerable to diverse threats, such as, power side-channel attacks, which infer sensitive data through dynamic power profile. In this paper, we present a novel framework, `POLARIS` for mitigating power side channel leakage using an Explainable Artificial Intelligence (XAI) guided masking approach. `POLARIS` uses an unsupervised process to automatically build a tailored training dataset and utilize it to train a masking model. The `POLARIS` framework outperforms state-of-the-art mitigation solutions (e.g., VALIANT) in terms of leakage reduction, execution time, and overhead across large designs.**

***Index Terms*—Power side-channel Analysis and Mitigation, Hardware Security, Explainable Artificial Intelligence.**

I. Introduction

Edge computing devices in Internet-of-Things (IoT) applications are increasingly used in diverse applications from domains, such as, Industry 4.0, surveillance, smart cities, defense, healthcare, and aerospace. These devices often record and process sensitive data and hence become a target for diverse attacks from malicious entities. A malicious entity can attempt to leak sensitive information from these devices through various means, such as timing side-channel attacks, power side-channel attacks, and fault-injection attacks.

In this work, we focus on mitigating power side-channel attacks [1], [2], which involve measuring and analyzing the power consumption traces of a system to infer the values of sensitive data. This inference is possible because different computing instructions consume different amount of power while running on an unprotected/unmasked electronic hardware. Wide array of techniques have been developed to mitigate this concern, which include:

1) **Quantifying Power Side Channel Leakage**: Techniques such as Test Vector Leakage Assessment (TVLA) were developed to estimate power side channel leakage at a gate level for a certain number of traces [3].
2) **Masking Gates**: Composite logic gates were developed to replace traditional gates to provide better power side channel leakage camouflaging [4], [5].
3) **Efficient Utilization of Masking Gates**: Brute force replacement of all gates using composite masking gates can lead to very high design overheads (area, power, delay). Hence, techniques such as VALIANT [6] were developed to perform selective masking to obtain a good masking effect at a lower overhead.

Existing power side-channel countermeasures, such as, [6], [7], [8], which relies on fixed heuristics (static algorithm), may not perform well for diverse designs. Additionally, relying on TVLA (a simulation-based approach) analysis makes it highly time-consuming to operate on large designs. DL [9] and LLM-based [10] leakage estimation techniques are proposed but are inadequate in leakage mitigation due to their high training times and lack of synthetic data support or explainability features. To mitigate these shortcomings, we present a novel Explainable AI based Design-for-Security framework `POLARIS` (POwer Side-Channel LeAkage Reduction using Explainable AI Solution), that utilizes a search process to automatically learn how to most optimally insert masking gates to achieve high power side channel leakage reduction at low overhead. We implement `POLARIS` with training data generated using smaller open-source designs and extensively evaluated its effectiveness across various designs with varying complexities. `POLARIS` is able to outperform VALIANT [6] in terms of performance (execution time), effectiveness (overall leakage reduction), and design overhead (area, power, delay). `POLARIS` does not rely on TVLA for leakage estimation and mitigation through masking and is approximately 6× faster than VALIANT, making it scalable for larger designs. In summary, we make the following key contributions:

1) Developed a novel explainable design-for-security (DFS) framework (`POLARIS`) for mitigating power side-channel vulnerabilities in electronic systems.
2) Designed an unsupervised algorithm to automatically generate training data for `POLARIS` using a search approach and an AI algorithm for efficiently masking a given digital design.
3) Implemented the `POLARIS` framework as a parameterized tool & integrated it into the ASIC design flow.
4) Extensively evaluated `POLARIS` (different settings) and compared it with VALIANT [6], a state-of-the-art alternative solution, using more than 10 large digital designs.

The rest of the paper is organized as follows: Section II describes the brief background and related work, followed by the motivation behind this work in Section III. Section IV outlines the methodology, and Section V demonstrates the experimental results. We conclude the paper in Section VI.

© This paper has been accepted at the 62nd Design Automation Conference (DAC), 2025

II. Background

A. Test Vector Leakage Assessment (TVLA)

TVLA [3] is a widely recognized method for power side-channel analysis based on Welch's t-test. The approach captures the DUT's leakage behavior under varied inputs to detect any differences. Mathematically, let us denote two sets of data by $\mathcal{Q}_0$ and $\mathcal{Q}_1$, their cardinality by n_0 and n_1, respectively. Let also μ_0 (resp. μ_1) and s_0^2 (resp. s_1^2) denote sample mean and sample variance of the set $\mathcal{Q}_0$ (resp. $\mathcal{Q}_1$). The t-test statistic and the degree of freedom v are computed as:

$$t = \frac{\mu_0 - \mu_1}{\sqrt{(\frac{s_0^2}{n_0}) + (\frac{s_1^2}{n_1})}} \qquad v = \frac{(\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1})^2}{\frac{(\frac{s_0^2}{n_0})^2}{n_0 - 1} + \frac{(\frac{s_1^2}{n_1})^2}{n_1 - 1}} \tag{1}$$

The t-statistic is typically assessed with a threshold of $\pm$ 4.5 for distinguishability. This threshold ensures that for $|t| > 4.5$ and $v > 1000$, the p-value drops below 0.00001, indicating over 99.999% confidence against the null hypothesis [11]. A crypto implementation is considered protected with 99.999% confidence if $|t| \leq 4.5$; otherwise, it may indicate leakage [6]. The test can be performed in two ways: (i) Fixed-vs-Fixed (using known intermediate values) or (ii) Fixed-vs-Random (using fixed and random input patterns) [12].

The naive algorithm to compute variance (s_0^2, s_1^2) followed by the respective standard deviation (ρ_0, ρ_1) is inefficient as it involves processing full trace points twice due to the dependency on mean (μ_0, μ_1) values for the computation. The evaluation for ρ_0 is given as:

$$\rho_0 = \frac{\sum_i (x_i - \mu_0)^2}{n_0}, \text{ where } x_i \in \mathcal{Q}_0 \tag{2}$$

TVLA trace collection is slow due to repeated mean and variance calculations. To accelerate it, [11] proposed an efficient one-pass method for raw and central moments computation during trace acquisition. The raw moment (sample mean) and central moment (variance) of order $d = 1$ for an expanded set $\mathcal{Q}' = \mathcal{Q} \cup y$ with n elements are calculated as:

$$M_1^{\mathcal{Q}'} = M_1^{\mathcal{Q}} + \frac{\Delta}{n}, \text{ where } \Delta = y - M_1^{\mathcal{Q}} \tag{3}$$

$$\mu = M_1 \text{ and } s^2 = CM_2, \text{ where } CM_2 = M_2 - M_1^2 \tag{4}$$

This method can be extended to compute the central moments at any $d > 1$ using the equations as given in [11].

B. Masking

Masking is a well-known countermeasure for mitigating side-channel attacks [13]. Masking randomizes sensitive data in cryptographic operations using a secret sharing scheme at the logic level. To achieve d-th order security, each variable x is split randomly into $(d + 1)$ shares, such that $x = x^{(1)} \oplus x^{(2)} \oplus \cdots \oplus x^{(d+1)}$, where $x^{(i)}$ denotes i^{th} share of x and $\oplus$ denotes XOR operation. This provides significant protection since that adversary would need to recover all $d+1$ shares to obtain the actual value of x, which requires statistical analysis up to d^{th} order. Authors in [4], [5] proposed efficient masking schemes for block cipher implementations against DPA attacks. As example, the *Masked AND* and *OR* operation proposed by [4] is demonstrated in Fig. 1 and described below:

If x_i and y_j denote the bits that mask the 'real' bits a_i and b_j then we denote masked bits as $\hat{a} = (a_i \oplus x_i)$ and $\hat{b} = (b_j \oplus y_j)$ correspondingly. We introduce a random bit, z, as a new mask for computing a *masked AND* operation as:

$$\mathcal{M}(a \cdot b) = ((\hat{a} \cdot \hat{b}) \oplus ((x_i \cdot \hat{b}) \oplus ((x_i \cdot y_j) \oplus z))) \oplus (y_j \cdot \hat{a}) \tag{5}$$

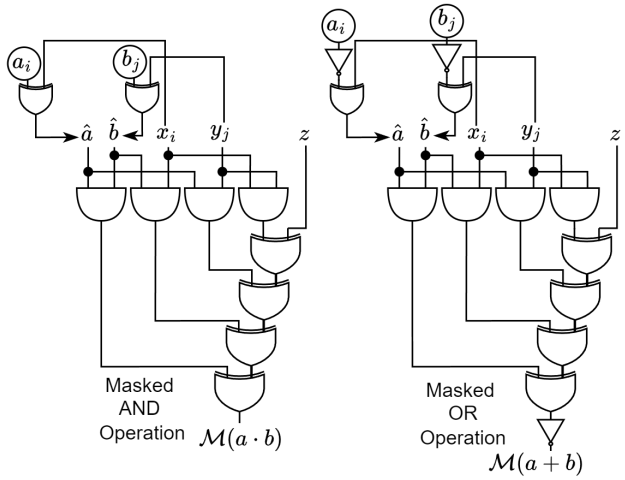


Fig. 1: *AND* and *OR* operation on masked data.

C. Related Work

Existing power side-channel leakage evaluation/mitigation approaches like CASCADE [7], Karna [8], VALIANT [6] incorporate TVLA analysis and hence suffers from high runtime and scalability issues. AI-based methods like DL-LA [9] and Netlist Whisperer [10] are often limited by the lack of training data and are vulnerable to adversarial attacks due to reliance on external training data [14], [15], [16], [17].

`POLARIS` stands out by using a synthetic data generation approach for creating its own training data, ensuring improved performance and resilience against adversarial attacks. `POLARIS` also incorporates XAI-based rules generation for efficient power side channel mitigation through masking. Table I provides a comparative analysis between the existing solutions and the proposed framework.

III. Motivation

A. Can Explainable AI Generate Custom DFS Rules/Models for Power Side Channel Mitigation?

Power side channel mitigation techniques such as Karna and VALIANT rely on static algorithms, in other words, a fixed set of hand-crafted rules. This is a limiting factor and prevents optimal operation on a diverse set of designs. We hypothesize that an Explainable AI (XAI) framework might be able to generate custom rules/models (to mitigate power side channel leakage) based on a given type of design, making the process more autonomous and adaptive.

B. Can We Bypass TVLA using AI?

Most of the prevalent power side channel mitigation frameworks (e.g., Karna, VALIANT) rely on TVLA for estimating

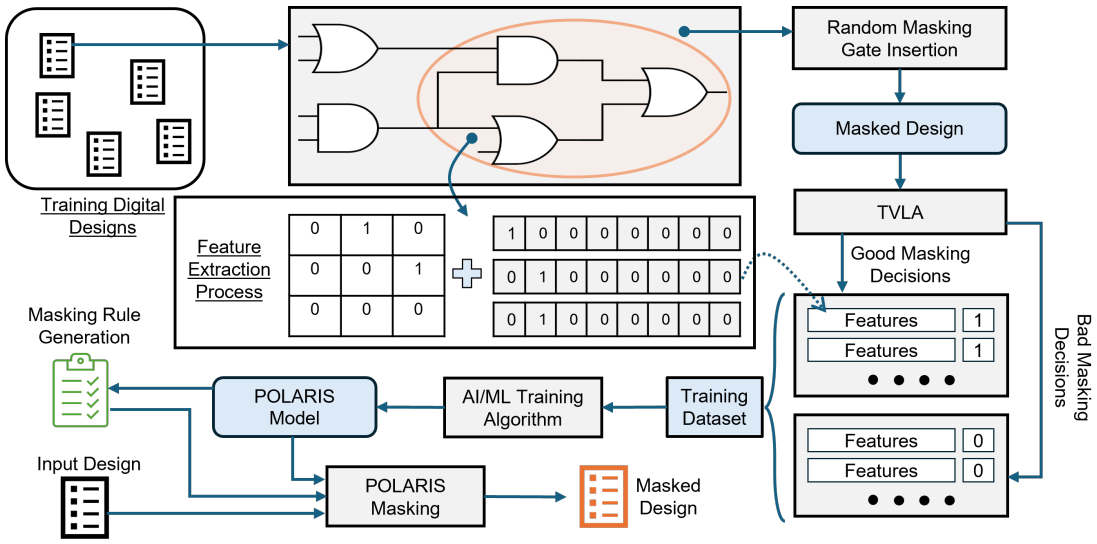


Fig. 2: Overview of the POLARIS framework.

power leakage. TVLA has major scalability concerns particularly for large industry-grade designs that in turn hampers the scalability of frameworks such as Karna and VALIANT. We hypothesize, AI techniques might be able to predict the leakage (directly or in an integrated way) associated with individual gates faster and with a linear time complexity (with respect to design size).

C. Can We Generate the Training Data?

Training data is a major limiting factor for AI frameworks in the hardware security domain. Techniques such as DL-LA [9] and Netlist Whisperer [10] require a very high amount of data making it almost infeasible to effectively scale across different design types. If we can develop a framework that can leverage synthetic training data automatically generated via an unsupervised technique, then it would be highly scalable across diverse design types.

IV. Methodology

We build on the motivations presented in Sec. III to design a framework that can: (1) Leverage XAI for creating custom power side channel mitigation rules/models; (2) Bypass TVLA using an AI approach; (3) Utilize automatically generated synthetic data for model training. The overall framework, shown in Fig. 2 operates in three major stages: (i) Knowledge extraction, data generation, and building the ML model; (ii) Interpreting the model using the explainable SHAP [18] framework; and (iii) Masking against side-channel vulnerabilities.

A. Knowledge Extraction and ML Model Development

Obtaining a large dataset for training AI models is a major concern particularly in the domain of hardware security and EDA. To address this concern, POLARIS utilizes a novel unsupervised approach to generate a large database for AI model training. As shown in Fig. 2, the framework POLARIS converts any digital design represented as gate-level netlist ($\mathcal{D}$) into a graph (G_r) such that $G_r = (V, E)$ where, V: gates and E: interconnections and randomly inserts masking gates based on mask size ($\mathcal{M}_{size}$). Then, POLARIS calculates leakage values ($\mathbb{L}_G$) using TVLA analysis and compares it to the original leakage values. If the difference exceeds a threshold (θ_r), the masking is labeled 'good' (1); otherwise, 'bad' (0) and appended to $\{X\_data, Y\_data\}$. These labels are linked to structural features used for knowledge extraction. The framework employs local structural features for training and evaluation, similar to [19],[20]. The structural features of a gate include information such as their local placement and interconnections. In a sub-design graph, gate connectivity

TABLE I: Comparing POLARIS with existing solutions for power side-channel leakage assessment and mitigation.

Approach	Method	Model Training		Feature Set	Mitigation Support	Performance	Platform
CASCADE [7]	TVLA	N/A		N/A	No	Slow	ASIC
Karna [8]	TVLA	N/A		N/A	Limited	Slow	ASIC
VALIANT [6]	TVLA	N/A		N/A	Yes	Slow	ASIC
DL-LA [9]	DL	Training Time: High Adversarial Attacks: Possible	Explainability: No Synthetic Data Support: No	Trace based	No	Slow	ASIC/ FPGA
Netlist Whisperer [10]	LLM	Training Time: High Adversarial Attacks: Possible	Explainability: No Synthetic Data Support: No	ANF equations	Yes	Slow	ASIC
POLARIS (This work)	XAI	Training Time: Low Adversarial Attacks: No	Explainability: Yes Synthetic Data Support: Yes	Structural Analysis	Yes	Fast	ASIC/ FPGA*

*POLARIS can be extended to FPGA design flow by re-training the model with lookup table (LUT) based FPGA netlists.

Algorithm 1: Cognition Generation

Input: $\mathcal{D}, \mathcal{M}_{size}, \mathcal{L}, itr, \theta_r$
Output: $\mathbb{M}$

$G_r \leftarrow$ ***graphify***$(\mathcal{D})$
$\mathbb{L}_G \leftarrow$ ***leak_estimate***$(\mathcal{D})$
$X\_data, Y\_data \leftarrow \emptyset$; $run \leftarrow 0$
$\mathcal{R}_{gates} \leftarrow G_r.gates$
while $\mathcal{M}_{size} \leq len(\mathcal{R}_{gates})$ *and* $run \leq itr$ **do**
  $\mathcal{S}_{gates} \leftarrow$ ***random***$(\mathcal{M}_{size}, \mathcal{R})$
  $\mathcal{D}_{mod} \leftarrow$ ***modify***$(\mathcal{S}_{gates}, \mathcal{D})$
  $\mathcal{R}_{gates} \leftarrow \mathcal{R}_{gates} - \mathcal{S}_{gates}$
  $\mathbb{L}_G^{mod} \leftarrow$ ***leak_estimate***$(\mathcal{D}_{mod})$
  for i ***in*** $\mathcal{S}_{gates}$ **do**
    $S_f \leftarrow$ ***structural_features***$(G_r, \mathcal{L}, i)$
    $r_{Ratio} \leftarrow$ ***compare***$(\mathbb{L}_G[i], \mathbb{L}_G^{mod}[i])$
    if $r_{Ratio} \geq \theta_r$ **then**
      $label \leftarrow 1$
    else
      $label \leftarrow 0$
    $X\_data.append(S_f)$
    $Y\_data.append(label)$
  $run \leftarrow run + 1$
$\mathbb{M} \leftarrow$ ***train_data***(X_data, Y_data)
return $\mathbb{M}$

is encoded with an adjacency matrix and one-hot encoding. Fig. 2 illustrates the structural feature extraction process, showing how gates are vectorized and how the dataset is created via feature extraction and labeling. Breadth-first search (BFS) is employed to explore neighboring gates (Locality $\mathcal{L}$) and assign labels based on leakage comparison. The random insertion process runs iteratively ($\leq itr$ times) until the termination condition is met. Finally, we obtain a trained model ($\mathbb{M}$) trained using $\{X\_data, Y\_data\}$ that will be used to generate human-understandable rules through model interpretability in the next stage. Algorithm 1 describes the sequence of steps involved in extracting knowledge through structural features and developing the ML model.

B. Explainable AI (XAI): Model Interpretability by SHAP

XAI enhances transparency and accountability in AI by providing methods for designing inherently interpretable models and for analyzing decisions after training. We incorporate the SHAP (SHapley Additive exPlanations) [18] algorithm into our framework to analyze the trained models and extract the underlying model rules driving the mitigation process. SHAP leverages game theory to quantify the contribution of each feature in model's prediction. For an individual prediction, the Shapley value of a feature f, is denoted as ϕ_f, and is computed using Equation 6. Given h total features, there are $h!$ possible permutations (coalitions) of these features. Let g represent a specific coalition that excludes feature f. The weighting factor $\frac{|g|!(h-|g|-1)!}{h!}$ accounts for the number of ways to form such coalitions, and the term $\text{val}(g \cup \{f\}) - \text{val}(g)$ signifies the

Algorithm 2: POLARIS Masking

Input: $\mathcal{D}, \mathbb{M}, \mathcal{R}_L, \mathcal{M}_{size}, \mathcal{L}$
Output: $\mathcal{D}_{\mathcal{M}}$

$G_r \leftarrow$ ***graphify***$(\mathcal{D})$
$\mathcal{G} \leftarrow G_r.gates$
$\mathcal{C} \leftarrow \emptyset$
for i ***in*** $\mathcal{G}$ **do**
  $S_f \leftarrow$ ***structural_features***$(G_r, \mathcal{L}, i)$
  $\mathcal{P}_{pred} \leftarrow$ ***inferencing***$(\mathbb{M}, R_L, S_f)$
  $\mathcal{C}.append(\mathcal{P}_{pred}, i)$
$\mathcal{C}_{top} \leftarrow$ ***sort_descending***$(\mathcal{C})$
$\mathcal{D}_{\mathcal{M}} \leftarrow$ ***modify***$(\mathcal{D}, \mathcal{C}_{top}, \mathcal{M}_{size})$
$\mathbb{L}_{info} \leftarrow$ ***leak_estimate***$(\mathcal{D}_{\mathcal{M}})$
return $\mathcal{D}_{\mathcal{M}}$

marginal contribution of feature f to coalition g. N denotes set of all features $\{1, 2, ...h\}$.

While there are other XAI frameworks, such as LIME [21] and Captum [22], we focus on SHAP due to its versatility. SHAP offers both model-agnostic methods (e.g., Kernel SHAP) and model-specific methods (e.g., Tree SHAP), making it suitable for a broad range of models. The automated rules, unlike handcrafted ones, can be used independently to make masking decisions or alongside the model to achieve better predictions and decisions, as shown in Fig. 2.

$$\phi_f = \sum_{g \subseteq N \setminus \{f\}} \frac{|g|!\,(h - |g| - 1)!}{h!} \left[\text{val}\left(g \cup \{f\}\right) - \text{val}(g)\right] \tag{6}$$

C. POLARIS Masking in Side-Channel Defense

The trained model can be used directly to mask a design for side-channel defense and can also leverage human-readable rules ($\mathcal{R}_L$) rules generated through the XAI framework to improve decision-making. For each gate in the graph (G_r), generated from ($\mathcal{D}$), structural features (S_f) are extracted, and predictions ($\mathcal{P}_{pred}$) are made using the model or rules followed by appending $\mathcal{P}_{pred}$ values to choices ($\mathcal{C}$). The set $\mathcal{C}$ is sorted in descending order to identify the top selections ($\mathcal{C}_{top}$). Finally, masking is applied to the design based on $\mathcal{C}_{top}$, and leakage of the masked design ($\mathcal{D}_{\mathcal{M}}$) is measured. Algorithm 2 outlines all the steps involved.

V. Results and XAI Analysis

In this section, we analyze the effectiveness of the proposed framework POLARIS in identifying and mitigating leakage from open-source designs with varying complexities and leveraging the SHAP framework to mitigate leakage through masking against power side-channel attacks.

A. Experiment Configuration

We utilize knowledge obtained from training on smaller designs to generalize to larger and entirely unseen designs, leveraging the transfer learning approach. The model is trained on smaller open-source designs that can adapt to larger designs, capturing patterns that can be transferred for practical

TABLE II: Quantitative comparison between VALIANT [6] & POLARIS in terms of leakage reduction & runtime efficiency.

Benchmarks	Leakage Value (Per Gate)					Total Leakage Reduction (%)				Time (s)	
	Before	VALIANT	POLARIS			VALIANT	POLARIS			VALIANT	POLARIS
			50%Mask	75%Mask	Full Mask		50%Mask	75%Mask	Full Mask		
des3	2.66	1.20	1.23	1.09	1.02	54.89	53.76	59.02	61.65	38.68	16.40
arbiter	1.42	1.10	1.15	0.98	0.88	22.54	19.01	30.99	38.03	97.62	15.11
sin	2.16	1.23	1.19	1.04	0.91	43.06	44.91	51.85	57.87	42.42	15.21
md5	2.93	1.39	1.41	1.28	1.21	52.56	51.88	56.31	58.70	202.11	59.11
voter	2.96	1.30	1.12	1.10	1.01	56.08	62.16	62.84	65.88	84.59	28.36
square	2.94	1.23	1.29	1.25	1.21	58.16	56.12	57.48	58.84	309.00	56.62
sqrt	3.04	1.44	1.48	1.31	1.20	52.63	51.32	56.91	60.53	281.05	75.53
div	2.28	1.44	1.37	1.28	1.18	36.84	39.91	43.86	48.25	389.26	92.51
memctrl	1.62	1.21	1.18	1.03	1.00	25.31	27.16	36.42	38.27	2257.50	94.44
multiplier	2.44	1.23	1.22	1.14	1.12	49.59	50.00	53.28	54.10	498.15	127.03
log2	2.25	1.16	1.20	1.11	1.06	48.44	46.67	50.67	52.89	428.31	148.38
Average	**2.43**	**1.27**	**1.26**	**1.15**	**1.07**	**45.46**	**45.72**	**50.88**	**54.09**	**420.79**	**66.25**

Note: **'X% Mask'** denotes X% of total number of leaky gates. The computational time for POLARIS with varying mask sizes is similar.

TABLE III: Comparison among different ML models used in POLARIS. Values indicate leakage reduction in %.

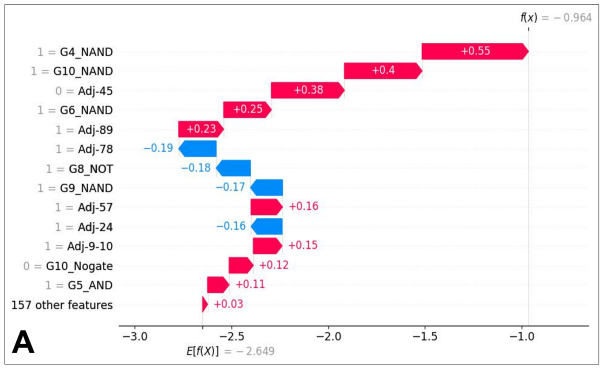
Designs	Random Forest	XGBoost	AdaBoost
des3	33.73	41.35	61.65
arbiter	29.36	43.02	38.03
sin	32.48	56.89	57.87
md5	43.97	54.70	58.70
voter	58.67	62.39	65.88
square	22.37	42.96	58.84
sqrt	39.02	53.57	60.53
div	68.69	74.36	48.25
memctrl	18.95	28.16	38.27
multiplier	57.09	55.09	54.10
log2	57.37	53.88	52.89
Average	**41.97**	**51.49**	**54.09**

Note: We observe nominal differences in computational time across models.

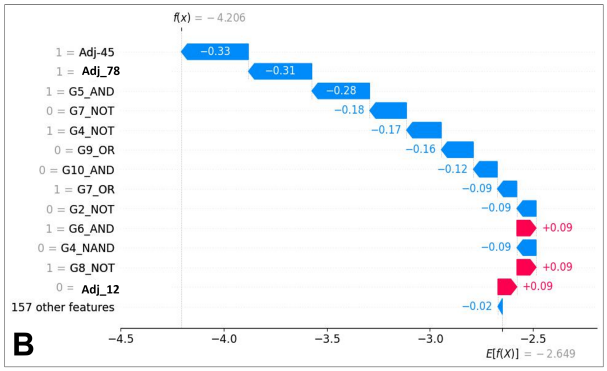
application of transfer learning. We used six designs from the ISCAS-85 benchmark suite [23] synthesized using Synopsys Design Compiler (DC) for training and simulated using 10,000 traces for leakage measurement by TVLA analysis. Selecting smaller designs helps reduce the model's training time (∼40 minutes). The key parameters are: $\mathcal{M}_{size} = 200$, $\mathcal{L} = 7$ (considering 7 neighboring gates), $itr = 100$, and $\theta_r = 0.70$ (indicating a leakage reduction of 70% or more as good masking). The rationale behind selecting the value of θ_r to 0.70 is that selecting higher values lead to significant data imbalance, which could cause the model to underfit and hinder its ability to generalize effectively. The evaluation phase includes larger and entirely different designs with varying $\mathcal{M}_{size}$ from EPFL [24] and MIT-CEP [25] benchmark suite that are distinct from training designs.

B. Comparing ML Models for Leakage Reduction

In this work, we explore several machine-learning techniques to develop improved masked designs that mitigate side-channel vulnerabilities. Table III compares the leakage reduction by Random Forest, XGBoost, and AdaBoost models for $\mathcal{L} = 7$, $\theta_r = 0.7$, and $\mathcal{M}_{size}$ denoting number of leaky gates identified by TVLA. We applied SMOTE [26] for Random Forest and employed weighted training for XGBoost and AdaBoost models for handling imbalance that occurred due to θ_r. The learning rate (α) for both XGBoost and AdaBoost was set to 0.01. AdaBoost outperforms the other models with a 54.09% leakage reduction on average, making it the chosen model for the remaining experiments.



(a)



(b)

Fig. 3: SHAP waterfall plots generated by POLARIS Adaboost (ADB) model for power side channel defense.

C. Comparison with State-of-the-Art

Table II compares the performance of VALIANT framework [6] and POLARIS in terms of leakage per gate, total leakage reduction, and execution time. POLARIS is tested with 50%, 75%, and full (100%) masking sizes. Even with only 50% masking, POLARIS achieves a 45.72% leakage reduction, surpassing VALIANT's 45.46% with full masking. At 75%

TABLE IV: Comparison of area, power, and delay overheads between VALIANT [6] and POLARIS[§].

Designs	Original			VALIANT (x Original[◇])			POLARIS[§] (x Original[◇])			Reduction (%) in POLARIS		
	Area (μM²)	Power (mW)	Delay (ns)	Area	Power	Delay	Area	Power	Delay	Area	Power	Delay
des3	9083.31	2.73	1.3	3.7	3.1	1.6	2.4	1.8	1.2	35.14	41.94	25.00
arbiter	10310.05	1.61	1.23	3.1	3.3	3.3	2.1	2	2.1	32.26	39.39	36.36
sin	13421.04	9.12	7.98	3.6	3.1	3.1	2.3	1.6	1.9	36.11	48.39	38.71
md5	24217.29	0.94	0.12	5.1	3.8	1.4	3.1	1.7	1.1	39.22	55.26	21.43
voter	28090.42	17.25	2.53	4.2	3.8	4.5	2.5	2.4	2.9	40.48	36.84	35.56
square	51117.09	32.98	9.84	4.7	3.8	3.2	2.7	2.3	1.2	42.55	39.47	62.50
sqrt	42921.71	83.28	279.99	5.3	3.8	3	3.2	2.3	1.9	39.62	39.47	36.67
div	44048.97	12.59	196.90	3.1	2.9	2.2	2.5	2.2	1.7	19.35	24.14	22.73
memctrl	42921.71	8.21	3.55	4	4.2	2.6	2.2	2.1	1.9	45.00	50.00	26.92
multiplier	66185.85	56.62	10.13	2.4	1.9	2	2.1	1.5	1.7	12.50	21.05	21.68
log2	77395.08	66.95	16.55	3.9	3.2	3.4	2.4	1.6	2.1	38.46	50.00	38.24
Average	**37246.59**	**26.57**	**48.19**	**3.92**	**3.35**	**2.75**	**2.50**	**1.95**	**1.79**	**34.61**	**40.54**	**33.25**

§ We utilize POLARIS w/ 50% Mask for comparison with VALIANT, achieving a comparable leakage reduction while masking half the number of gates.
◇ Overheads are reported as x times original value.

TABLE V: Power side-channel mitigation rules generated via the POLARIS framework (AdaBoost Model).

Rules	As long as	Procedure
A	G4 = NAND && G5 = AND && G4 (NAND) and G5 (AND) are not connected && G6 = NAND && G10 = NAND && G8 (NOT) and G9 (NAND) are connected	Select & Replace with masking gate
B	G5 = AND && G4 = NOT && G7 = OR && G4 (NOT) and G5 (AND) are connected blue && G7 (OR) and G8 (NOT) are connected	Do not Mask

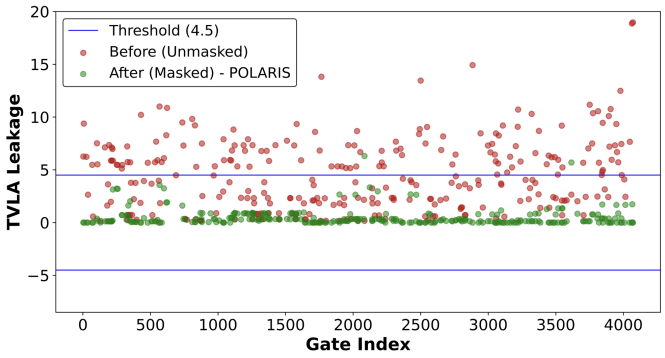


Fig. 4: TVLA values before and after masking in *des3* design. Gates exceeding threshold (±4.5) are considered as *leaky*.

and full masking, POLARIS reduces leakage by 50.88% and 54.09%, respectively, outperforming VALIANT. Additionally, POLARIS operates 6× faster than VALIANT, demonstrating its efficiency in leakage mitigation. Table IV compares area (μM^2), power (mW), and delay (ns) overheads of the unmasked design (Original) with the respective masked designs produced by VALIANT and POLARIS. As we can observe, POLARIS reduces area by 34.61%, power by 40.54%, and delay by 33.25% compared to VALIANT while also achieving greater leakage reduction. These results highlight the effectiveness of POLARIS in mitigating side-channel vulnerabilities with reduced resource overhead.

D. Explainability: POLARIS Extracted Rules

Using SHAP algorithms, we interpret the decision-making process for individual samples by examining waterfall plots in Fig. 3, which illustrate the contribution of each feature to the prediction of the target variable. Here, x represents a selected observation, $f(x)$ is the model's predicted value for this input, and $E[f(x)]$ denotes the expected value of the target variable, essentially the average prediction across all observations. The length of each bar in the plot indicates the SHAP value (Fig. 3). POLARIS can extract rules for leakage reduction; these rules are summarized in Table V.

E. Additional Analysis & Discussion

To visualize the effectiveness of POLARIS (see Fig. 4), we performed TVLA analysis for all gates in the *des3* design *before* (●) and *after POLARIS masking* (●). The performance of POLARIS can be improved by including a wider variety of designs in the training process, enabling POLARIS to better comprehend design diversity. POLARIS can also be extended to support other masking gates (e.g., DOM [5]), providing a flexible and robust solution for addressing power side-channel leakage mitigation with improved performance.

VI. Conclusion

Power side-channel leakage of sensitive on-chip data is a major security threat. Existing mitigation frameworks face scalability issues (limited data, TVLA reliance, slow runtime) and lack effectiveness. This paper presents the POLARIS, a DFS framework that can be integrated into commercial ASIC design flow to generate design-specific rules using XAI for low-overhead power side-channel mitigation. POLARIS is fast and utilizes a synthetic data generation scheme to bypass data-related concerns associated with most AI frameworks. We observe significant performance gain over state-of-the-art solutions in terms of speed, leakage reduction, and design overhead. Future works will make the POLARIS framework more efficient and extend it to other side-channels (EM/timing).

VII. Acknowledgment

This material is supported by the National Science Foundation (NSF) under Grant No. 2350363 and Grant No. 2316399.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[2] A. Moradi, "Side-Channel Leakage through Static Power – Should We Care about in Practice?" Cryptology ePrint Archive, Paper 2014/025, 2014. [Online]. Available: https://eprint.iacr.org/2014/025

[3] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A testing methodology for side channel resistance validation," in *NIST Non-Invasive Attack Testing Workshop*, vol. 7, 2011, pp. 115–136.

[4] E. Trichina, "Combinational Logic Design for AES SubByte Transformation on Masked Data," Cryptology ePrint Archive, Paper 2003/236, 2003. [Online]. Available: https://eprint.iacr.org/2003/236

[5] H. Gross, S. Mangard, and T. Korak, "Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order," in *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, ser. TIS '16, New York, NY, USA, 2016, p. 3.

[6] R. Sadhukhan, S. Saha, S. Paria, S. Bhunia, and D. Mukhopadhyay, "VALIANT: An EDA Flow for Side-Channel Leakage Evaluation and Tailored Protection," *IEEE Transactions on Computers*, vol. 73, no. 2, pp. 436–450, 2024.

[7] D. Šijačić, J. Balasch, B. Yang *et al.*, "Towards efficient and automated side-channel evaluations at design time," *J Cryptogr Eng*, vol. 10, pp. 305–319, 2020.

[8] P. Slpsk, P. K. Vairam, C. Rebeiro, and V. Kamakoti, "Karna: A Gate-Sizing based Security Aware EDA Flow for Improved Power Side-Channel Attack Protection," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

[9] T. Moos, F. Wegener, and A. Moradi, "DL-LA: Deep learning leakage assessment: A modern roadmap for SCA evaluations," Cryptology ePrint Archive, Paper 2019/505, 2019. [Online]. Available: https://eprint.iacr.org/2019/505

[10] M. Nair, R. Sadhukhan, H. Pearce, D. Mukhopadhyay, and R. Karri, "Netlist whisperer: Ai and nlp fight circuit leakage!" in *Proceedings of the 2023 Workshop on Attacks and Solutions in Hardware Security*, ser. ASHES '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 83–92. [Online]. Available: https://doi.org/10.1145/3605769.3623989

[11] T. Schneider and A. Moradi, "Leakage Assessment Methodology - a clear roadmap for side-channel evaluations," Cryptology ePrint Archive, Paper 2015/207, 2015. [Online]. Available: https://eprint.iacr.org/2015/207

[12] O. Reparaz, "Detecting Flawed Masking Schemes with Leakage Detection Tests," in *Revised Selected Papers of the 23rd International Conference on Fast Software Encryption - Volume 9783*, ser. FSE 2016. Berlin, Heidelberg: Springer-Verlag, 2016, p. 204–222.

[13] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *Advances in Cryptology – EUROCRYPT 2013*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 142–159.

[14] P. Pathmanathan, S. Chakraborty, X. Liu, Y. Liang, and F. Huang, "Is poisoning a real threat to llm alignment? maybe more so than you think," *arXiv preprint arXiv:2406.12091*, 2024.

[15] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer security–ESORICs 2020: 25th European symposium on research in computer security, ESORICs 2020, guildford, UK, September 14–18, 2020, proceedings, part i 25*. Springer, 2020, pp. 480–501.

[16] W. Guo, B. Tondi, and M. Barni, "An overview of backdoor attacks against deep neural networks and possible defences," *IEEE Open Journal of Signal Processing*, vol. 3, pp. 261–287, 2022.

[17] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *arXiv preprint arXiv:1912.02771*, 2019.

[18] S. Lundberg, "A unified approach to interpreting model predictions," *arXiv preprint arXiv:1705.07874*, 2017.

[19] P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine learning guided structural analysis attack on hardware obfuscation," in *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 2018, pp. 56–61.

[20] P. Chakraborty, J. Cruz, R. Almawzan, T. Mahfuz, and S. Bhunia, "Learning Your Lock: Exploiting Structural Vulnerabilities in Logic Locking," *IEEE Design & Test*, 2024.

[21] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you? Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[22] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan *et al.*, "Captum: A unified and generic model interpretability library for pytorch," *arXiv preprint arXiv:2009.07896*, 2020.

[23] J. Hayes, "ISCAS Benchmark Circuits for Logic Synthesis and Verification," accessed: 2024-11. [Online]. Available: https://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html

[24] EPFL-LSI, "The EPFL Combinational Benchmark Suite," accessed: 2024-11. [Online]. Available: https://www.epfl.ch/labs/lsi/page-102566-en-html/benchmarks/

[25] MIT-LL, "CEP: Common Evaluation Platform," accessed: 2024-11. [Online]. Available: https://github.com/mit-ll/CEP/

[26] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.