IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2026, No. 2, pp. 441–467. DOI:10.46586/tches.v2026.i2.441-467

AIPS: AI-Based Power Simulation for Pre-Silicon Side-Channel Security Evaluation

Ya Gao[1], Haocheng Ma[1], Tanchen Zhang[2], Jiaji He[1], Yiqiang Zhao[1], Mirjana Stojilović[3] and Yier Jin[4]

[1] Tianjin University, Tianjin, China,
gaoyaya@tju.edu.cn,hc_ma@tju.edu.cn,dochejj@tju.edu.cn,yq_zhao@tju.edu.cn
[2] Hong Kong Polytechnic University, Hong Kong, China, tanchen.zhang@connect.polyu.hk
[3] EPFL, Lausanne, Switzerland, mirjana.stojilovic@epfl.ch
[4] University of Science and Technology of China, Hefei, China, jinyier@ustc.edu.cnu

Abstract.
Power side-channel analysis (SCA) attacks pose a significant threat to the security of integrated circuits, making pre-silicon evaluation essential for vulnerability identification. Existing approaches face a fundamental trade-off: commercial electronic design automation (EDA) tools can generate accurate nanosecond-resolution transient power traces, but become prohibitively slow as design size and trace length increase; in contrast, recent artificial intelligence (AI) methods largely target average-power estimation and overlook fine-grained transients critical to SCA. To address this gap, we propose AIPS, an AI-based gate-level power simulation framework that uses a diffusion model to synthesize transient power traces conditioned on switching activity and technology-library features. AIPS is trained on power traces generated by Synopsys PrimeTime PX (PTPX) and evaluated on five representative targets (AES, Kyber, two masked AES variants, and a RISC-V core) using waveform-similarity metrics and first- and higher-order SCA. Results show that AIPS matches PTPX closely under waveform metrics and reproduces the same evaluation outcomes as PTPX under the tested attack settings, while delivering 4.14×–42.44× speedup over PTPX when scaling to 1M traces. In inference-only mode, AIPS achieves up to 10k× speedup. AIPS maintains accuracy with small training sets (about 1k traces or less), and scales favorably with increasing trace length, enabling large-scale pre-silicon side-channel security evaluation.

Keywords: Power Side-Channel Analysis · Pre-Silicon · EDA · Diffusion Model

1 Introduction

Side-channel analysis (SCA) attacks were introduced by Kocher et al. in the 1990s [KJJ99]. SCAs exploit various forms of information leakage, such as timing, power consumption [ZLYW23, UJP24, SGS23], and electromagnetic (EM) emissions [BBJP19], generated by physical devices during operation to extract sensitive information within circuits. Despite continued advances in attack methodologies, power SCA remains a primary practical threat due to simple measurement equipment, stable signal strength, and mature data-processing pipelines. As a result, effective power side-channel evaluation has become a critical step in hardware security assurance.

Current industry practice predominantly performs SCA post-silicon, which creates two fundamental limitations. First, Application Specific Integrated Circuit (ASIC) fabrication is costly and time-consuming. Once a side-channel vulnerability is identified, the iterative cycle of mitigation, tape-out, and re-evaluation becomes prohibitively expensive. To

Received: 2025-10-15 Accepted: 2025-12-15 Published: 2026-04-23

Table 1: Related work in power consumption prediction.

Work	Simulation Level	Simulation Granularity	Model	Functional Verification
Woudenberg et al. [VWGV+23]	Netlist	Transient	LUT	SCA (AES)
Zhang et al. [ZRSK22]	Netlist	Average	GPU	DU[1] (NVDLA open source project)
Lu et al. [LZX24]	RTL	Average	Linear	DU (RISC-V)
Nasser et al. [NSP+20]	RTL	Average	MLP	DU (Arithmetic module)
Lorandel et al. [LPH16]	Netlist	Average	MLP	DU (Baseband processing IP)
Zhou et al. [ZRZ+19]	RTL	Average	CNN	DU (Arithmetic units, NoC router, RISC-V)
Zhang et al. [ZRK20]	Netlist	Average	GCN	DU (Arithmetic units, NoC router, RISC-V)
Rakesh et al. [RDTA23]	RTL	Average	GNN	DU (iscas-85, combinational benchmark)
Aljuffri et al. [ASR+23]	Netlist	Transient	GAN	SCA (AES)
This Work	**Netlist**	**Transient**	**Diffusion**	**SCA (AES, Kyber, masked AES, RISC-V)**

reduce costs, Field-Programmable Gate Arrays (FPGAs) are commonly used for early prototyping and security evaluation. However, FPGA implementations differ substantially from ASICs in architecture, routing, and physical layout, thereby limiting the extent to which side-channel observations transfer to final silicon. Second, post-silicon evaluation provides limited visibility into the origin of leakage, making it difficult to design targeted countermeasures [MNBV23]. In this context, pre-silicon evaluation emerges as a promising alternative: it enables design-time security evaluation and can be integrated into mainstream EDA flows [PPFT22, ŠBY+20].

A key obstacle is efficiency. Mainstream power simulation tools such as PTPX and Voltus are widely adopted in standard design flows, yet their runtime becomes a critical bottleneck when used for pre-silicon SCA evaluation [LS24, YMZ+15]. For example, Sadhukhan et al. reported that generating a single gate-level transient power trace for a PRESENT circuit using PTPX takes 5.47 seconds [SMRM19]. Security evaluations often require tens of thousands to millions of traces, thus making direct use of numerical power engines impractical at scale.

Recent work has therefore explored fast power prediction. As summarized in Table 1, existing methods fall into three broad paradigms: (i) Look-up table (LUT)-based approaches that precompute state-dependent power for gates and generate traces directly from Value Change Dump (VCD) activity [VWGV+23]; (ii) GPU-accelerated simulators that parallelize event-driven power computation [ZRSK22]; and (iii) AI-driven models that learn power from activity or structure, ranging from Linear model [LZX24], multilayer perceptron (MLPs) [NSP+20, LPH16], convolutional neural network (CNNs) [ZRZ+19], and graph neural network (GNN) [ZRK20, RDTA23] regressors to generative models [ASR+23]. However, for pre-silicon side-channel evaluation, existing work still falls short along several practical axes that jointly determine whether an approach is usable in an SCA setting. Effective pre-silicon SCA evaluation imposes the following requirements:

1) Simulation level: Achieving a balance between accuracy, speed, and flexibility for design iterations is crucial in power simulation, and the netlist emerges as an attractive compromise [SS23, HPN+19]. It captures timing and process-node effects beyond the register-transfer level (RTL) while avoiding layout costs and supporting rapid iteration.

2) Simulation granularity: Nanosecond-resolution transient power analysis is fundamental for SCA. Average power may suffice for conventional design verification, but it obscures data-dependent fluctuations that are critical for leakage detection.

3) Model construction: The simulation model must reflect the fundamental mech-

[1]DU = Design updates (only in Table 1).

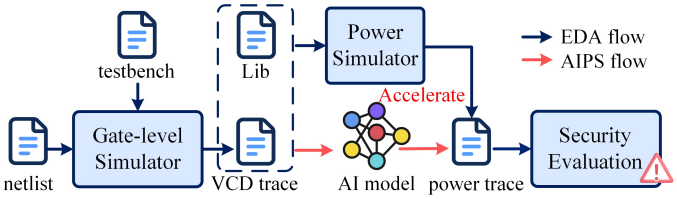


Figure 1: The AIPS flow vs. the EDA flow.

anisms of power consumption, scale to large netlists, and generate stable traces. LUT-based methods cannot fully capture variations from signal switching and load fluctuations [VWGV+23]. Linear power models fail to capture complex patterns of power consumption influenced by nonlinear factors [LZX24]. Generic MLP or CNN approaches often require extensive handcrafted encodings, whereas GNNs align with netlist structure but scale poorly with circuit size and signal count, thereby limiting scalability [FNS24]. Overly rich feature sets risk the curse of dimensionality, while excessively complex models overfit and generalize poorly. Generative adversarial networks (GANs) add stochasticity but are prone to training instability and mode collapse [SGZ+16].

4) Functional verification: Evaluation should span diverse circuits and protection schemes. Many studies focus solely on AES, leaving capability boundaries unclear.

5) Simulation accuracy: High-fidelity traces must match not only waveform shape but also preserve side-channel information. Latest methods still show noticeable shape gaps and weak similarity scores [ASR+23], which can lead to false positives or misleading conclusions [ABPP21, KLS22].

Taken together, these requirements imply more than a fast predictor: pre-silicon SCA needs a scalable trace generator that can synthesize nanosecond-resolution, netlist-level transient power while retaining rare, data-dependent transients. In practice, the five axes above can be reduced to three concrete needs: high trace fidelity; preservation of leakage-relevant variability; and stable training with simple deployment. To meet these needs, we consider several learning paradigms, from deterministic regression to stochastic generative models, and identify *diffusion* as the best match for the SCA setting.

Classical regression-based approaches learn a deterministic mapping from activity features to a single output trace and therefore tend to approximate an average response [HTF09]. This often produces overly smooth waveforms that fail to capture conditional variability and rare transient patterns, both of which are critical for evaluating higher-order leakage. GANs alleviate this by injecting stochasticity, but their training is widely recognized as unstable and prone to mode collapse; prior work has also observed degraded temporal alignment and spectral fidelity in generated traces [ASR+23].

Diffusion models provide a better fit to the needs above. First, their iterative denoising process supports reconstruction of fine-grained transient details, improving trace fidelity [LLF+25]. Second, diffusion models are likelihood-based generative models trained by optimizing a variational bound on the data log-likelihood, empirically yielding stable training and strong mode coverage [SDME21]. This helps retain subtle, data-dependent variations critical for SCA, rather than collapsing into a single dominant behavior. Third, the training objective is fixed (nonadversarial), avoiding GAN instabilities and making optimization and deployment straightforward [Viv24]. Moreover, diffusion models have been successfully applied to time-correlated waveform generation (e.g., audio [KPH+20, CZZ+20]), suggesting that they are well-suited to synthesizing transient traces.

Therefore, we propose a pre-silicon power simulation approach in which a diffusion model serves as the core trace-generation engine, acting as a surrogate for the numerical power analysis engine. We realize this approach in AIPS, an AI-based power simulator trained on transient power traces generated by PTPX. Figure 1 contrasts the AIPS flow with the standard EDA flow. We summarize our contributions as follows:

- We develop AIPS, a gate-level diffusion-based framework for pre-silicon transient power trace generation. Unlike diffusion-based SCA studies that operate on post-silicon traces for denoising or augmentation [KPP24, YJ24], AIPS uses diffusion as a surrogate power-analysis engine.
- AIPS includes dedicated parsers for power-related features: a Lib parser that distills power LUTs and load capacitance into cell-level embeddings; a VCD parser that slices activity into fixed time windows and builds per-window toggle-count dictionaries (nanosecond resolution); and a power-trace parser that extracts and aligns PTPX transient power to the same windows.
- We compare AIPS with PTPX and the state-of-the-art GAN baseline [ASR+23], using PTPX traces as ground truth. On AES, AIPS achieves substantially lower DTW/PSD-distance, and in inference-only mode reaches up to 10k× speedup vs. PTPX and 60× vs. GAN. Under the same time budget, AIPS generates 4.15× more traces than PTPX.
- We evaluate AIPS across diverse targets to explore its capability boundaries: standalone encryption IPs (AES and Kyber), two masked AES variants, and a RISC-V CPU. Through comprehensive first- and higher-order SCA, our results demonstrate that AIPS preserves leakage-relevant information and supports reliable evaluation.
- Lastly, we release AIPS as open source, to support future work [GMZ+26].

In the remainder of this paper, Section 2 provides foundational knowledge on CMOS power consumption principles and the theoretical framework of diffusion models. Section 3 details the AIPS architecture, including its key components and workflow. Sections 4, 5, and 6 present experimental results of AIPS applied to encryption IPs, masked designs, and a RISC-V CPU. Section 7 analyzes both training performance and minimum training budget. Section 8 analyzes the efficiency and scalability of AIPS compared to traditional EDA tools. Section 9 concludes the paper and discusses future work.

2 Background

2.1 CMOS Power Consumption

Modern digital CMOS circuits exhibit two fundamental power dissipation mechanisms, as implemented in Synopsys' PTPX [Nes18]. The first is leakage power (P_{leakage}), which occurs due to subthreshold conduction and reverse-biased diode currents when transistors are in steady-state conditions. P_{leakage} is calculated as the product of supply voltage (V_{dd}) and the aggregate leakage current (I_{leakage}) as shown in Equation 1. The second mechanism, dynamic power consumption, occurs during logic transitions and consists of internal power (P_{internal}) and switching power ($P_{\text{switching}}$). P_{internal} accounts for energy dissipation within the logic cell boundaries during switching events. This includes short-circuit current (I_{sc}) that flows momentarily when both NMOS and PMOS transistors are simultaneously active during input transitions, as well as the energy required to charge or discharge internal node capacitance (C_{int}). Equation 2 captures these effects, where the first term represents short-circuit power (P_{sc}) and the second term models internal switching power (P_{intsw}) proportional to the toggle rate (T_{r}).

$$P_{\text{leakage}} = V_{\text{dd}} \times I_{\text{leakage}} \tag{1}$$

$$P_{\text{internal}} = P_{\text{sc}} + P_{\text{intsw}} = V_{\text{dd}} \times I_{\text{sc}} + \tfrac{1}{2} \times C_{\text{int}} \times V_{\text{dd}}^2 \times T_{\text{r}}. \tag{2}$$

Switching power ($P_{\text{switching}}$) in Equation 3 dominates the dynamic power consumption in most designs, representing the energy needed to charge the external load capacitance

(C_{load}) through the output driver. PTPX calculates these power components using switching activity information from VCD or Switching Activity Interface Format (SAIF) files and detailed cell power models from technology libraries. The library models contain multidimensional look-up tables that capture the complex dependencies of leakage currents and internal power on input transition times, output loads, and environmental conditions such as voltage and temperature.

$$P_{\text{switching}} = \frac{1}{2} \times C_{\text{load}} \times {V_{\text{dd}}}^2 \times T_{\text{r}} \tag{3}$$

2.2 Diffusion Model

The diffusion model proposed by Ho et al. in 2020 [HJA20] represents a major advance in generative modeling. As a member of the deep generative family, diffusion models stand out for a noise-based generation mechanism and, when properly optimized, can surpass GANs in sample quality while remaining more stable [DN21]. These advantages have led to successful deployments in large-scale vision systems (e.g., DALL·E2 [RPG+21], Imagen [SCS+22]).

Diffusion models comprise two processes (see Figure 2): forward noise diffusion and backward denoising. The forward process is a Markov chain that perturbs clean samples $\boldsymbol{x}_0$ into near-Gaussian $\boldsymbol{x}_T$; the backward process learns to reverse this corruption.

1) Forward process: The forward process incrementally adds Gaussian noise to the original samples $\boldsymbol{x}_0$, producing a series of noisy samples $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$ after successive noise superposition, where T denotes the total number of diffusion steps. By carefully controlling the noise-addition parameters, the forward diffusion step ensures that the original samples gradually degrade to almost pure noise samples $\boldsymbol{x}_T$ after T steps. The process is defined in Equation 4.

$$q\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}\right) := \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1},\, \beta_t \boldsymbol{I}\right) \tag{4}$$

$\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$ represent the noisy samples at diffusion steps $t-1$ and t, respectively. At step t, the model does not generate a sample arbitrarily. Instead, it samples around the previous step's sample by slightly scaling down $\boldsymbol{x}_{t-1}$ and adding Gaussian noise. The parameter β_t controls the magnitude of the noise added at diffusion step t, where $\beta_t \in (0, 1)$ for $t = 1, 2, \ldots, T$. The sequence $\{\beta_t\}_{t=1}^{T}$ defines the noise schedule over the T diffusion steps. $\boldsymbol{I}$ denotes the identity matrix with the same dimension as the input sample. $\mathcal{N}$ represents the Gaussian distribution of $\boldsymbol{x}_t$, whose mean is $\sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1}$ and whose variance is $\beta_t \boldsymbol{I}$.

2) Backward process: The backward process p_θ consists of a series of noise reduction steps, where a neural network is trained to incrementally remove noise from a noisy sample until it is restored to a clean sample. Here, θ represents the parameters of the denoising network. The process is described by Equation 5:

$$p_\theta\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right) := \mathcal{N}\left(\boldsymbol{x}_{t-1}; \mu_\theta\left(\boldsymbol{x}_t, t\right), \epsilon_\theta\left(\boldsymbol{x}_t, t\right)\right) \tag{5}$$

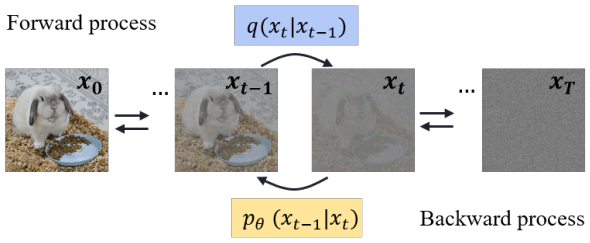


Figure 2: The forward and backward processes of a diffusion model.

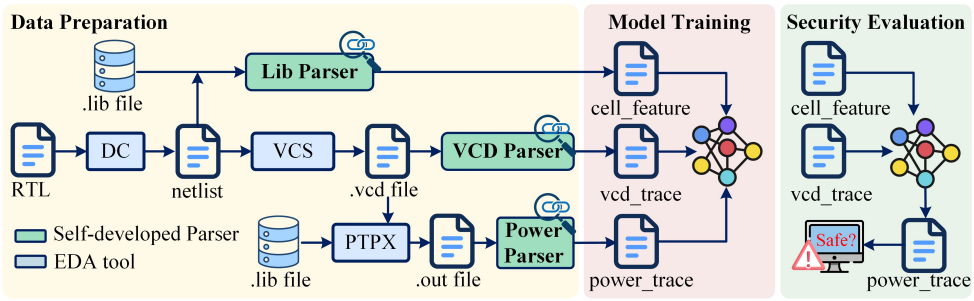


Figure 3: The proposed AIPS framework.

$\mu_\theta(\boldsymbol{x}_t, t)$ and $\epsilon_\theta(\boldsymbol{x}_t, t)$ represent the mean and variance of the reverse process, respectively. During the inference phase, samples from the true distribution can be generated by first sampling from the noisy distribution and then applying a step-by-step denoising process over T steps.

Owing to their ability to generate high-quality, diverse samples, diffusion models achieve state-of-the-art performance in image generation, translation, and inpainting [DN21, SCC+22, LDR+22]. They have also been explored in the SCA domain: Karayalcin et al. pioneered their use for denoising captured side-channel traces by leveraging the model's inherent noise-removal capabilities [KPP24]. Yap et al. employed mask data as labels to train the diffusion model, enabling the generation of additional traces from noise for security evaluation [YJ24]. In this work, our diffusion model takes fused VCD and cell-level features as input and iteratively refines a noisy latent representation into power traces, enabling fast trace generation for efficient pre-silicon side-channel security evaluation.

3 AI-based AIPS Framework

As shown in Figure 3, AIPS comprises three phases: data preparation, model training, and security evaluation. During the data preparation phase, Design Compiler (DC) synthesizes the RTL design into a gate-level netlist. Meanwhile, the Lib parser extracts critical power characteristics from the technology library (.lib file) into a structured *cell_feature* file. The Verilog Compiled Simulator (VCS) performs functional simulation to generate VCD files that capture precise signal switching activities, which are subsequently processed by the VCD parser into time-aligned *vcd_trace* matrices. PTPX then executes power simulations, with results stored in the *power_trace* file by the power parser, serving as ground-truth values for AIPS. During training, a diffusion model is trained to learn the complex nonlinear mapping from circuit activity to power-consumption patterns. The final security evaluation phase leverages the trained model to generate accurate power traces, enabling efficient side-channel evaluation without the computational overhead of EDA tools.

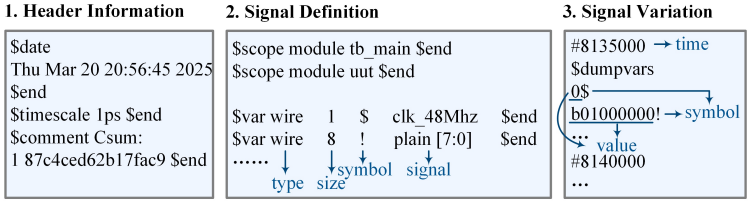


Figure 4: The composition of a .vcd file.

Algorithm 1: Lib_Parser (Lib → cell_feature)

Input: Liberty file $\mathcal{L}$ at target PVT
Output: Dictionary $\mathcal{F}$: cell_type ↦ cell_feature $\mathbf{f}_c$

Parse $\mathcal{L}$ into cells and pins;
foreach *cell* c **do**
  $L \leftarrow c$.`cell_leakage_power`;
  $\mathcal{E} \leftarrow [\,]$; `// internal-power summary`
  foreach *output pin* p *in* c **do**
    $C_{\text{pin}} \leftarrow p$.`capacitance`;
    foreach *internal_power of* p *(per related_pin)* **do**
      $E^{\uparrow} \leftarrow$ median(`rise_power.values`);
      $E^{\downarrow} \leftarrow$ median(`fall_power.values`);
      append $\langle E^{\uparrow}, E^{\downarrow} \rangle$ to $\mathcal{E}$;
  $\mathbf{f}_c \leftarrow \text{normalize}\big(L,\ C_{\text{pin}},\ \mathcal{E}\big)$;
  $\mathcal{F}[c.\texttt{name}] \leftarrow \mathbf{f}_c$;
return $\mathcal{F}$

	0~1ns	1~2ns	2~3ns	…	159~160ns
Signal_1	1	1	1	…	1
Signal_2	3	3	3	…	3
Signal_3	1	1	1	…	1
Signal_4	0	0	0	…	0
	…	…	…	…	…
Signal_i	2	2	2	…	2
Mean	0.018	0.014	0.010	…	0.046

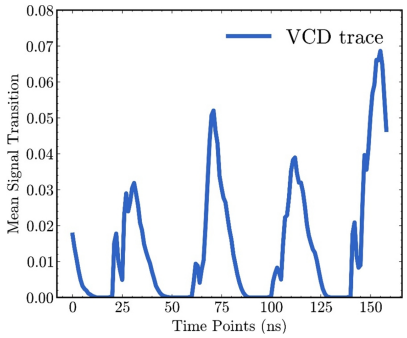


Figure 5: An example of parsing signal counts with corresponding VCD trace.

3.1 Data Preparation

To support diffusion-model training, we develop dedicated parsers for three key design files (.lib, .vcd, and .out), converting raw data into structured representations that capture essential power characteristics and are compatible with our model.

1) Lib Parser: The .lib file provides timing and power characterization for standard cells at specific process corners. Our parser extracts cell-level features that map directly to the power formulation in Section 2.1 (see Algorithm 1). For P_{leakage}, we extract them directly. For P_{internal}, PTPX queries `rise_power` and `fall_power` tables indexed by transition values via interpolation. Instead of retaining full lookup tables, we summarize them into representative intermediate values (median samples), avoiding dimensionality. Cell-driven effective load capacitance is completed later through netlist connectivity. This generates a dictionary-style cell-feature record indexed by cell type, providing lightweight, stable, low-dimensional embeddings for power-trace generation.

2) VCD Parser: The .vcd file (Figure 4) records signal switching activity as time-stamped value changes. Our parser converts long waveforms into stimulus-aligned, fixed-length windows of per-signal toggle counts at 1 ns resolution, serving as a proxy for T_{r} in Equation 2. As summarized in Algorithm 2, the parser operates in two stages. First, the original VCD is sliced into n per-stimulus files by preserving the header, extracting fixed time windows, and resetting the time base so that each slice starts at 0. This yields aligned VCD files with identical window length ℓ and resolution $\Delta t = 1\,\text{ns}$. Second, for a selected module, the parser reconstructs the symbol–signal mapping from the header

Algorithm 2: VCD parser: stimulus-aligned 1-ns toggle counts

Input: Long VCD $\mathcal{V}$, header path H, module name M, start time t_0, window length ℓ, time interval t_{in}, time resolution Δt=1 ns, stimuli n
Output: `vcd_trace` $\in \mathbb{R}^{n\times\ell}$

`/* Cutting (per-stimulus slices) */`
for $k \leftarrow 0$ **to** $n-1$ **do**
  write header from $\mathcal{V}$ to slice k;
  copy events with timestamps in $\left[t_0 + k\cdot t_{\text{in}},\ t_0 + \ell + k\cdot t_{\text{in}}\right)$;
  rebase time by subtracting $\left(t_0 + k\cdot t_{\text{in}}\right)$ so the first stamp is 0;
`/* Parsing and counting */`
parse header H to build symbol↔signal map under module M;
for $k \leftarrow 0$ **to** $n-1$ **do**
  build time resolution sequence $[0, \Delta t)$, $[\Delta t, 2\Delta t)$, … up to ℓ for slice k;
  for *each sequence s in slice k* **do**
    for *each event line within s* **do**
      if *single-bit* **then**
        increment that signal's toggle count;
      else if *multi-bit* **then**
        compare adjacent bit-vectors and increment per-bit counts on changes;
  aggregate per-signal counts and average; store as row k of `vcd_trace`;
return `vcd_trace` (size $n \times \ell$);

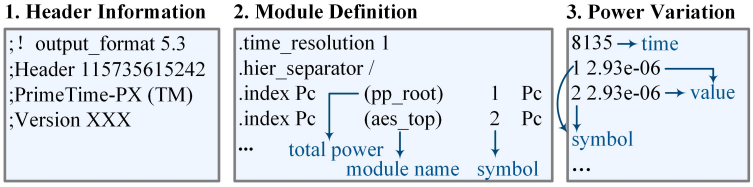


Figure 6: The composition of an .out file.

and counts per-signal value changes within each $[t,\ t+\Delta t)$ interval. Multi-bit signals are expanded into individual bits and counted at the bit level. The resulting n stimuli are aggregated into *vcd_trace* as an $n \times \ell$ array by averaging selected signal counts (Figure 5) . Optional GPU support is provided for parallel processing, producing identical outputs.

3) Power Parser: PTPX reports transient power in .out format (Figure 6), comprising a header, symbolic module indices (assign index numbers to modules based on their hierarchy), and time-stamped hierarchical power records. Given the simulation start time t_0, window length ℓ, time resolution Δt=1 ns, and number of stimuli n, our parser selects either total-power or module-scoped series, and assembles a time-windowed power matrix of size $n \times \ell$. The result is stored as *power_trace*.

3.2 Model Construction

Our AIPS diffusion framework is illustrated in Figure 7. It is built from shallow MLP blocks for simplicity and speed [PWP22]. The model first concatenates and projects two drivers of CMOS power: (i) *vcd_trace* and (ii) *cell_feature*, into a compact embedding via a small Fusion-MLP. This embedding forms the clean latent, $\boldsymbol{x}_0$, to which we add noise during the diffusion process. To model power as a distribution conditioned on activity and technology, we perturb $\boldsymbol{x}_0$ with a short Gaussian noise schedule (linear β) to obtain noisy latent $\boldsymbol{x}_t$. This "noise addition" step is a training mechanism that encourages the model to learn a denoising map and, consequently, the conditional distribution of transient power in digital circuits, given *vcd_trace* and *cell_feature*. The encoder then extracts

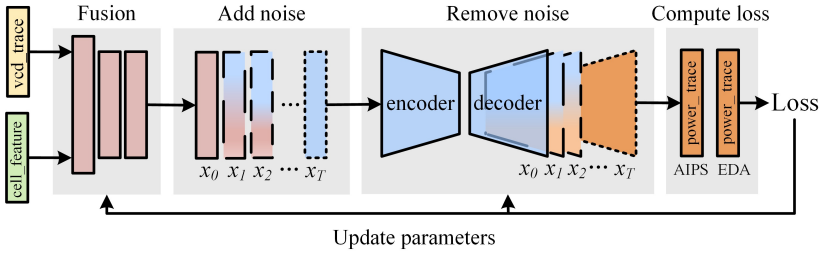


Figure 7: The diffusion model architecture in AIPS.

Table 2: Hyperparameters of the diffusion-inspired encoder–decoder model.

Hyperparameter	Fusion/Encoder	Decoder
Model parameters	Fusion input (1131+160, 256) ReLU() Fusion output (256, 256) ReLU() Encoder layer (256, 128) ReLU()	Input layer (128, 128) ReLU() Output layer (128, 160) Linear
Training Parameters	Noise level Noise steps Epochs Optimizer Loss function Learning rate	$[10^{-3}, 10^{-2}]$ 10 300 Adam MSE 0.001

Note: Noise steps correspond to the number of diffusion steps t, i.e., the number of β_t values, while the noise level defines the range of β_t in the schedule.

power-relevant features from the noise-perturbed samples while filtering out nonessential information, and the decoder reconstructs the transient power trace at the target output dimensionality from these encoded representations.

The training pipeline of AIPS follows standard practices in diffusion models, with adaptations to the computational and data constraints of power trace generation. To train and evaluate AIPS, we generate N labeled traces per design using PTPX. We split these labeled traces into 90% for training and 10% for testing. Separately, we generate an additional independent validation set of size N with disjoint stimuli, used only for early stopping and model selection. All input features are standardized using zero-mean and unit-variance normalization computed from the training set. The model is trained for up to 300 epochs, with early stopping triggered when the validation loss does not improve for 10 consecutive epochs, to prevent overfitting.

Table 2 summarizes the hyperparameter settings, illustrated with power traces of length 160 and cell features of dimension 1131. Since different designs may involve different cell sets or trace lengths, changes in feature dimensionality require retraining. The Fusion-MLP uses a hidden dimension of 256; the encoder compresses the latent representation to 128 dimensions, and the decoder restores it to the output resolution. Compared to smaller alternatives (e.g., 48 or 96), this configuration consistently achieves a favorable speed–accuracy trade-off. The diffusion process employs a linear Gaussian noise schedule. As the focus is on short transient power windows, a small number of diffusion steps is sufficient. We use 10 steps with β_t uniformly spaced in the range $[10^{-3}, 10^{-2}]$. Training is performed using the Adam optimizer with a learning rate of 10^{-3}. Adam is widely used due to its robustness to gradient noise and its ability to handle heterogeneous feature scales [KB15, HJA20]. The model is optimized by minimizing the mean squared error (MSE) between the generated power traces and the PTPX reference traces. MSE is a

standard choice in diffusion-based waveform synthesis, as it directly optimizes sample-level fidelity in the time domain and provides stable training behavior [ASS24].

AIPS is implemented in Python 3 using PyTorch libraries. Although the framework supports GPU acceleration, all performance evaluations are conducted on the CPU only to ensure a fair comparison with PTPX, which does not use GPUs. Unless otherwise stated, all experiments are performed on an Intel Xeon Gold 6230 CPU with a base frequency of 2.1 GHz running Red Hat Enterprise Linux 7.6 and equipped with 256 GiB of DRAM.

3.3 Security Evaluation Metrics

Evaluating the performance of diffusion models in the context of SCA presents a multifaceted challenge. Drawing on the frameworks established in [ASR+23, BBYS22], we classify the evaluation metrics into two categories. The first category includes dynamic time warping (DTW), cosine similarity (CS), and power spectral density (PSD), which are employed to evaluate the visual fidelity of the generated traces.

1) DTW is an algorithm designed to quantify the similarity between two temporal sequences by aligning them optimally to minimize the overall distance. In this study, we use the FastDTW [SC07, ASR+23] algorithm, employing the Euclidean distance for DTW computation. Smaller distances indicate greater similarity between the real and generated traces.

2) PSD quantifies the power distribution across different frequency components of a signal [SM+05, ASR+23]. The PSD traces of the real and generated traces closely match, suggesting that the generated trace captures a similar frequency profile.

3) CS evaluates the similarity between two vectors by computing the cosine of the angle between them. Unlike Euclidean distance, CS emphasizes the directional agreement of vectors. A CS value closer to 1 indicates higher similarity between the vectors.

However, the visual similarity between the generated and real traces does not inherently guarantee their applicability to side-channel security evaluation. Consequently, the second category of evaluation metrics focuses on the SCA metrics, specifically the test vector leakage assessment (TVLA) and the correlation power analysis (CPA) [MNBV23].

4) TVLA is an SCA method based on hypothesis testing, which uses Welch's t-test to compare two sets of traces with fixed and random plaintexts [GJJR11, SM15, PGA+23]. If the difference between the two sets of traces exceeds a predetermined threshold, it indicates the presence of leakage. Welch's t-test metric is calculated using Equation 6:

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{\sigma_0^2}{n_0} + \frac{\sigma_1^2}{n_1}}} \tag{6}$$

where n_i, μ_i and σ_i^2 stand for the sample size, mean, and variance, respectively.

5) CPA uses Pearson correlation for key disclosure, measuring the effectiveness of the generated traces through correlation coefficients [KJJ99] and the minimum trace to disclosure (MtD) [THH+05]. This approach provides a more concrete assessment of the security implications of the generated traces. Specifically, the correlation coefficient in Equation 7 is calculated between the predicted Hamming values $h_{d,i}$ and the measured power samples $t_{d,j}$ across D total measurements (or "stimuli," referring to individual encryption operations or data inputs):

$$r_{i,j} = \frac{\sum_{d=1}^{D}(h_{d,i} - \overline{h}_i) \cdot (t_{d,j} - \overline{t}_j)}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \overline{h}_i)^2 \cdot (t_{d,j} - \overline{t}_j)^2}} \tag{7}$$

Here, i indexes different hypothetical intermediate values (often corresponding to key hypotheses), while j indexes time points (or sampling points) within a power trace. The overlined terms $\overline{h}_i$ and $\overline{t}_j$ denote the average values of $h_{d,i}$ and $t_{d,j}$ over all D measurements,

respectively. In SCA, the Hamming model usually refers to the Hamming weight (HW) or Hamming distance (HD) model, which estimates power consumption by assuming that power consumption is related to the number of "1" bits in the processed data or the number of bits that change from "0" to "1" (or vice versa).

4 AIPS vs. EDA Flow on Unprotected Encryption Designs

In this section, we present the prediction results of AIPS through security evaluations of unprotected AES and Kyber designs. Our evaluation asks the following questions:

- How does AIPS perform in terms of accuracy on various experimental circuits compared to the EDA method (PTPX)?
- Can the traces generated by AIPS effectively support side-channel security evaluation?

4.1 Experimental Setup

All designs are synthesized at a SMIC 180 nm technology node. We create a dataset comprising 1k traces for training and testing (90%/10% split), and another 1k for validation. The input features vcd_trace and $cell\_feature$ are extracted by parsers described in Section 3. The ground-truth $power\_trace$ is obtained from PTPX; the setup is summarized in Table 3. For AIPS performance analysis experiments, we create a new evaluation dataset containing vcd_trace and $cell\_feature$, which the trained AIPS model then uses to predict power traces.

Table 3: Parameters for power simulation.

Design	Number of logic cells	Clock frequency	Resolution	Trace length
AES	11,509	25 MHz	1 ns	160 ns, 4 clock cycles
Kyber	126,527	50 MHz	1 ns	160 ns, 8 clock cycles

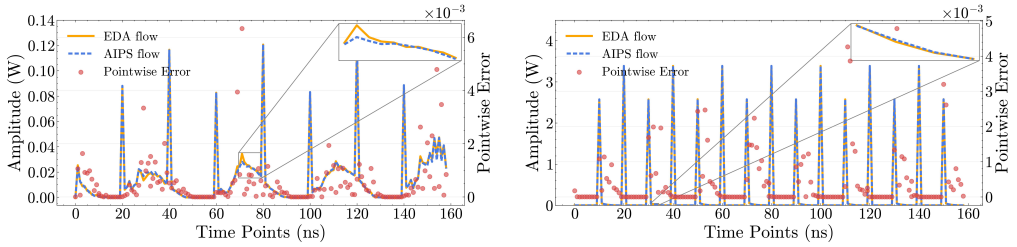
1) AES: implements the complete AES algorithm, designed according to the NIST standard for 128-bit plaintext and key. Within the trace window, AES executes the SubBytes operation of the first encryption round, which is a typical leakage source. This operation spans four consecutive clock cycles, with four S-Box operations performed per cycle, corresponding to four expanded key bytes processed in parallel.

2) Kyber: implements the decryption function of the Kyber-768 algorithm where each private key contains 256 12-bit coefficients. Kyber-768 is a version of the Crystals-Kyber algorithm with a security level of 3 [RRD+23]. Kyber uses shift registers to implement the Encode and Decode functions, and two sets of butterfly unit (BU) modules to implement the Compress, Decompress, number theoretic transform (NTT), inverse NTT, and point-wise multiplication (PWM) functions. We restrict the trace window to the execution segment dominated by the point-wise multiplication operation between the polynomial vector $\mathbf{u}$ (ciphertext) and the private key $\mathbf{s}$ in the NTT domain, which is a well-known leakage source in Kyber implementations [MPG+22, ZPM+23]. Since the BU carries out this computation, we retain only the .vcd and .out files corresponding to the BU module during data preparation.

4.2 Accuracy Performance Analysis

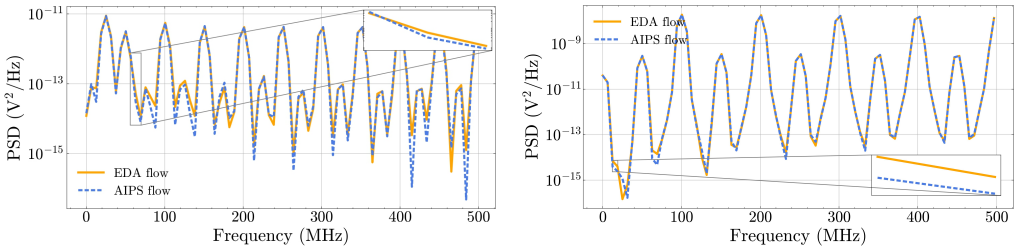
4.2.1 Visual Accuracy

For each circuit, we simulate 1k power traces with PTPX and predict the corresponding 1k traces with AIPS under the same VCD stimuli, enabling one-to-one comparison. For each trace pair, we compute DTW, PSD, and CS, and report the mean values in Table 4



(a) AES traces on evaluation dataset. (b) Kyber traces on evaluation dataset.

Figure 8: Trace comparison in time domain.



(a) AES traces on evaluation dataset. (b) Kyber traces on evaluation dataset.

Figure 9: Trace comparison in frequency domain.

to quantify AIPS's visual fidelity. Accuracy values for the GAN baseline are taken from the original paper [ASR+23].

Figure 8 visualizes time-domain similarity using DTW: the EDA trace (yellow) and the AIPS trace (blue) are overlaid for the same stimuli, with red dots marking pointwise errors. Figure 9 presents the frequency-domain comparison via PSD. Compared to the GAN baseline in [ASR+23] on AES, AIPS significantly reduces DTW, from 0.50 to 0.074 (6.7×).

4.2.2 Side-channel Accuracy

We perform TVLA and CPA to quantify the presence of side-channel information in the traces. For TVLA, we follow the standard Welch's t-test (fixed vs. random plaintext), using two sets of 5k traces each with an empirical threshold of 4.5. For CPA, the evaluation dataset contains 1k traces.

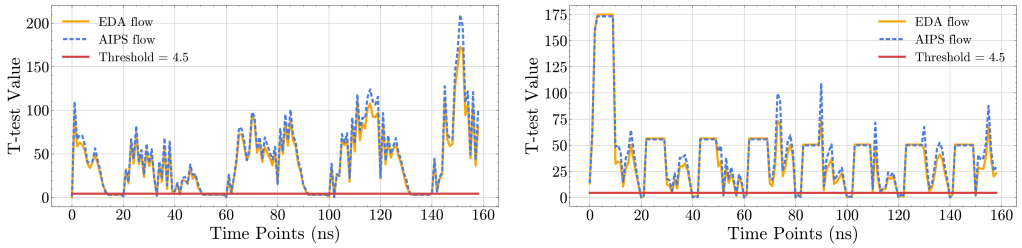
1) AES: Figure 10 (a) indicating that the time ranges for potential key leakage are concentrated in the first half of each clock cycle following the rising edge, specifically within 20−40 ns, 60−80 ns, 100−120 ns, and 140−160 ns.

Further CPA targeting the state registers of the S-Box module, we build the related HD model as the leakage model. CPA results on training and evaluation datasets are shown in Table 5, including the number of key bytes successfully recovered, the average of the maximum correlation values, and the average of the MtD. Both EDA traces and AIPS traces successfully recover the complete key and achieve similar side-channel metrics. For

Table 4: Accuracy comparison results under visual metrics.

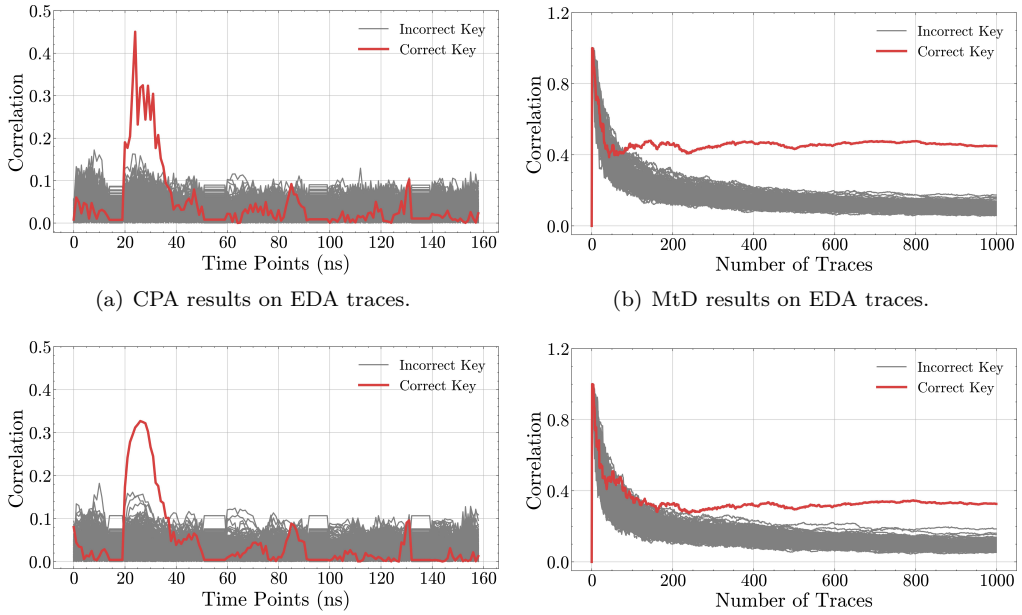
Design	DTW	PSD	CS
AES	0.0744	2.5877×10^{-14}	0.9995
Kyber	0.0850	6.2314×10^{-13}	0.9999
AES [ASR+23]	0.5000	1.4000×10^{-6} *	Not mentioned

* Estimated based on the image data in the paper.



(a) AES. (b) Kyber.

Figure 10: TVLA results of AES and Kyber.



(a) CPA results on EDA traces. (b) MtD results on EDA traces.

(c) CPA results on AIPS traces. (d) MtD results on AIPS traces.

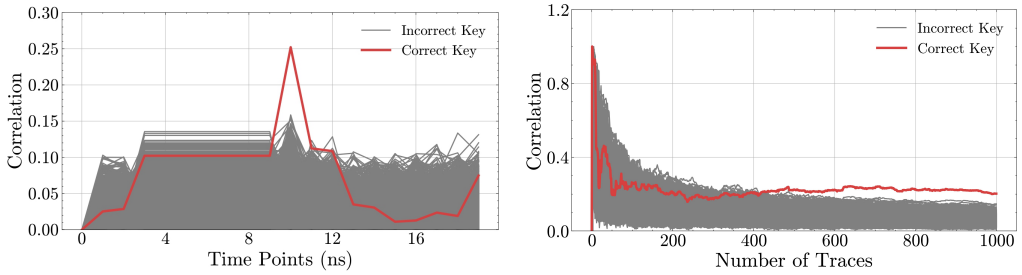
Figure 11: CPA results of AES on evaluation dataset.

a more intuitive understanding, Figure 11 presents the correlation results and MtD results of AES's first byte of the key on evaluation datasets. The correlation trace for the correct key byte exhibits a distinct peak (located within $20-40$ ns, coinciding with the TVLA results) and converges rapidly within around 140 traces.

2) Kyber: For Kyber-768, the PWM function consists of three rounds of multiplication operations between $\mathbf{s}[j]$ and $\mathbf{u}[j]$, denoted as $j = 0, 1, 2$. By constructing different $\mathbf{u}$ values to influence the computation results, we can sequentially recover the coefficients of $\mathbf{s}[0]$, $\mathbf{s}[1]$, and $\mathbf{s}[2]$. During each clock cycle, the BU module performs parallel multiplications between $\mathbf{u}$ and two coefficients of $\mathbf{s}$. Based on the BU architecture, we select an eight-cycle window in which the first 16 coefficients of $s[j]$ are processed (two coefficients per cycle).

Table 5: Side-channel accuracy metrics of AES and Kyber.

	AES				Kyber			
	Training		Evaluation		Training		Evaluation	
	EDA	AIPS	EDA	AIPS	EDA	AIPS	EDA	AIPS
Recovered key byte	1–16	1–16	1–16	1–16	3–16	3–16	3–16	3–16
Average correlation	0.4239	0.3795	0.4250	0.3701	0.2120	0.2038	0.2171	0.1907
Average MtD	111	140	107	141	327	441	348	415



(a) CPA results on EDA traces.

(b) MtD results on EDA traces.

(c) CPA results on AIPS traces.

(d) MtD results on AIPS traces.

Figure 12: CPA results of Kyber on evaluation dataset.

The TVLA results in Figure 10 (b) reveal that private key coefficient leakage occurs in every clock cycle. Then, we use the HD model to extract the coefficients of $\mathbf{s}[j]$. The HD model is constructed as follows:

$$L(i) = \alpha \times \mathrm{HD}((\mathbf{s}^T \circ \mathbf{u})[i]) + N \tag{8}$$

where i represents the leakage information from the i-th operation, α is a scaling factor and $\mathbf{s}^T$ stands for s in NTT Domain. HD(x) denotes the Hamming distance between the current and previous data, and N represents noise.

Table 5 summarizes the attack results for the first 16 coefficients of $\mathbf{s}[0]$ in the first round. Notably, neither the EDA traces nor the AIPS traces successfully recover the first two coefficients of $\mathbf{s}[0]$. This is because, for these coefficients, the attacker cannot obtain the HD value from previous time steps and must instead rely on the HW model, thereby significantly increasing the attack's difficulty. Figure 12 presents the attack results for the third coefficient of $\mathbf{s}[0]$ in the first round, evaluated on the evaluation dataset. The computation of this coefficient occurs within the $20-40$ ns time window. Both the EDA flow and the AIPS flow exhibit similar correlation peaks within this interval, and the correct key is distinguished from the incorrect keys after approximately 430 traces. Across AES and Kyber, AIPS reproduces the exact leakage locations (TVLA) and yields comparable key-recovery behavior (CPA/MtD) under the evaluated leakage models.

5 AIPS vs. EDA Flow on Masked Encryption Designs

In our hypothesis, once designers identify security vulnerabilities in the design, they iteratively refine the design by incorporating safeguards and reevaluating its security. To illustrate this process, we use AES as an example and enhance its security by adding masking. This approach not only demonstrates the iterative design improvement process but also tests the capability boundaries of AIPS. Our evaluation aims to answer the following questions:

- How well does AIPS support side-channel security evaluation of advanced masking schemes?

- Can AIPS reveal vulnerabilities caused by flawed mask implementations?

5.1 Experimental Setup

We evaluate two masked AES designs synthesized in the SMIC 180 nm node: AES_mask_o (first-order fixed masking) and AES_mask_d (second-order random masking). The model training steps align with Section 4.1, and the simulation parameters are summarized in Table 6. For leakage evaluation, protected designs typically require more traces than unprotected ones.

Table 6: Parameters for power simulation.

Design	Number of logic cells	Clock frequency	Resolution	Trace length
AES_mask_o	149,123	10 MHz	1 ns	160 ns, ∼2 clock cycles
AES_mask_d	51,695	25 MHz	1 ns	160 ns, 4 clock cycles

1) AES_mask_o: implements the well-known classical masking scheme proposed by Oswald et al. [OMPR05], using a combination of additive and multiplicative masks, balancing security with minimal area and time overhead. The trace window covers approximately two clock cycles, capturing the masked input ($plaintext \oplus key \oplus mask$) and its corresponding masked S-Box output.

2) AES_mask_d: implements the second-order masking scheme proposed by Dhooghe et al. [DSM22], leveraging three-share secret splitting, ring refreshing, and guard chaining to achieve full second-order security with only 1268 bits of fresh randomness while matching traditional three-share implementations in area and latency. This design enters a steady state (i.e., starting from the second encryption round) after reset deassertion. We focus on the four clock cycles during the internal operations of the masked S-box: masked GF(16) multiplication, the formation of nonlinear intermediate variables, and GF(16) inversion, which constitute the sensitive stages at which leakage is most likely to occur.

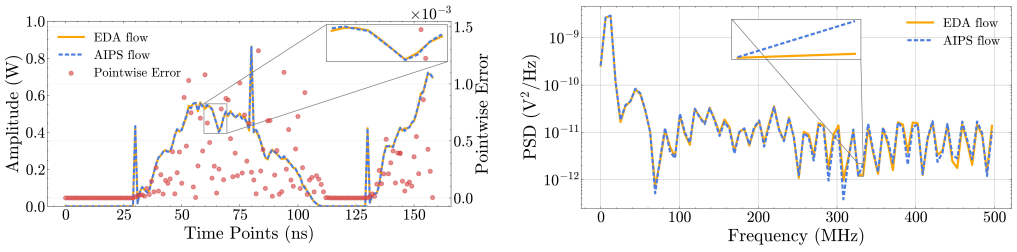
5.2 Accuracy Performance Analysis

5.2.1 Visual Accuracy

Table 7 reports AIPS visual metrics results over 1000 averages. Time- and frequency-domain comparisons are shown in Figures 13 and 14.

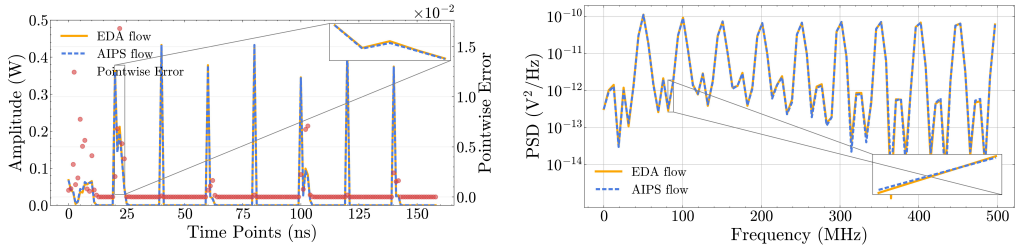
Table 7: Accuracy comparison results under visual metrics.

Design	DTW	PSD	CS
AES_mask_o	0.0322	6.3256×10^{-14}	0.9999
AES_mask_d	0.0712	9.9651×10^{-14}	0.9999



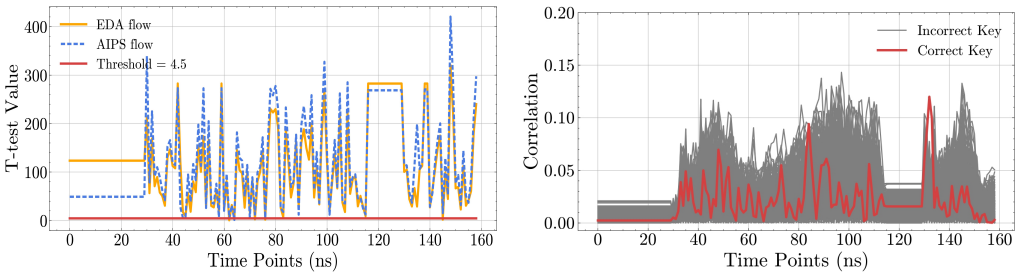
(a) Time domain. (b) Frequency domain.

Figure 13: Trace comparison of AES_mask_o on evaluation dataset.



(a) Time domain. (b) Frequency domain.

Figure 14: Trace comparison of AES_mask_d on evaluation dataset.



(a) TVLA results. (b) First-order CPA results.

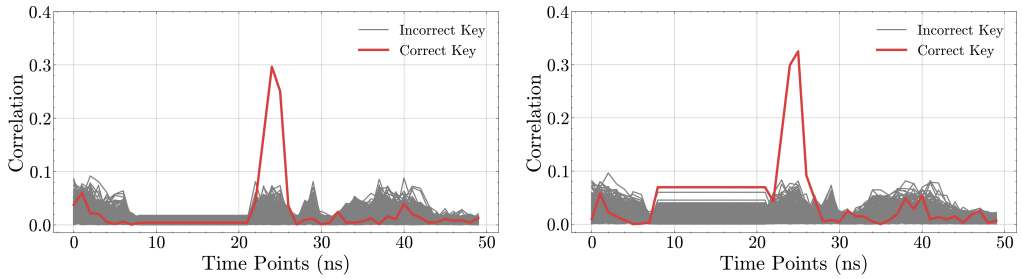
Figure 15: TVLA and First-order CPA results of AES_mask_o.

5.2.2 Side-channel Accuracy

1) AES_mask_o: We perform TVLA using 5k fixed and 5k random plaintext traces generated by AIPS. As shown in Figure 15(a), many T-values exceed the empirical threshold of 4.5, indicating first-order leakage. EDA traces show the same behavior, confirming that the observed leakage is not an artifact of AIPS. This behavior is consistent with prior observations that Oswald's masking scheme, while algorithmically secure, remains vulnerable in hardware due to glitches [MPO05].

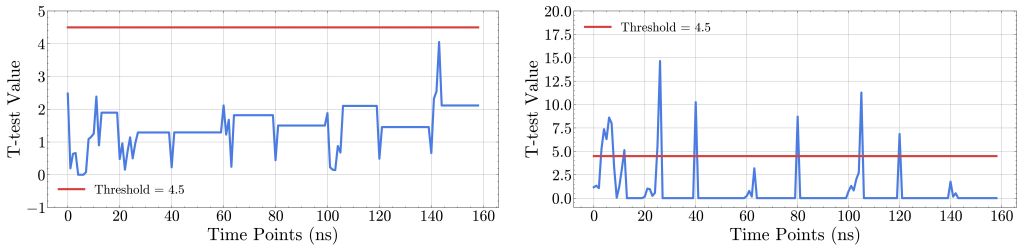
Guided by TVLA, we then run first-order CPA (illustrated for the first key byte in Figure 15(b)). The correlation for the correct key byte is only slightly higher than that for the wrong correlation traces, and the difference is not statistically significant. In our setup, only eight key bytes are clearly recoverable at first order, motivating a second-order attack. When the original S-Box (S) is replaced by a masked S-Box (S'), the relationship $S'(X \oplus M) = S(X) \oplus M$ holds for the input X and a fixed mask M. Our attack targets the first round of encryption, focusing on the traces of two critical operations [OMHT06]. The first trace corresponds to $P \oplus K \oplus M$, where P is the plaintext and K is the key. The second trace corresponds to the output of the masked S-Box: $S'(P \oplus K \oplus M) = S(P \oplus K) \oplus M$. To perform the second-order CPA, we compute the absolute difference between the two traces: $|C(S(P \oplus K) \oplus M) - C(P \oplus K \oplus M)|$ and conduct a correlation analysis with the model $HW(S(P \oplus K) \oplus (P \oplus K))$, where HW denotes the Hamming weight and C denotes the power consumption traces. This approach allows us to exploit potential second-order leakage arising from interactions among masked intermediate values. Figure 16 presents the second-order CPA attack results for the first key byte using EDA and AIPS-generated power traces. The maximum correlation for the correct key is approximately 0.3, which clearly distinguishes it from incorrect keys, and all key bytes can be recovered.

2) AES_mask_d: For the more complex masked scheme, we use AIPS to generate 400k traces, including 200k fixed and 200k random plaintext. Following the evaluation methodology of Dhooghe et al. [DSM22], we perform first-, second-, and third-order TVLA on these traces. The TVLA results shown in Figure 17(a) indicate that all T-values for second-order analysis remained significantly below the empirical threshold of 4.5 under



(a) Second-order CPA results on EDA traces. (b) Second-order CPA results on AIPS traces.

Figure 16: Second-order CPA results of AES_mask_o.



(a) 2nd-order TVLA. (b) 3rd-order TVLA.

Figure 17: TVLA results of AES_mask_d.

our test setup, indicating that no second-order leakage is detected in these traces. The third-order analysis, shown in Figure 17(b), reveals partial leakage consistent with the original findings reported by [DSM22].

6 AIPS vs. EDA Flow on Processor Design

Unlike standalone encryption IPs, we evaluate AIPS in a processor-design setting. The goal is not only to assess trace fidelity but also to examine whether AIPS preserves activity-dependent power characteristics at the full-core level, enabling side-channel security evaluation under pipelined execution.

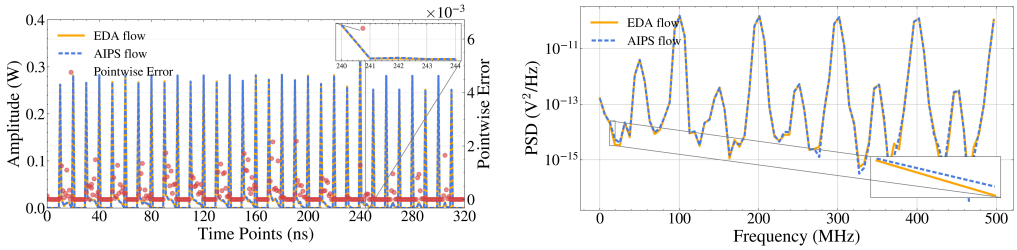
6.1 Experimental Setup

Our target design is derived from an open-source 32-bit in-order RISC-V core [Cav24]. We extend the instruction set architecture with AES instructions and apply lightweight optimizations following prior work [TG06, Saa20]. The design is synthesized in a 180 nm technology at a clock frequency of 50 MHz and contains 11,660 logic cells. Each AES encryption round executes a fixed sequence of 16 instructions. We analyze a 320 ns window (16 clock cycles) which fully covers the pipelined execution of the first round. Data preparation and model training follow descriptions in Section 4.1.

6.2 Accuracy Performance Analysis

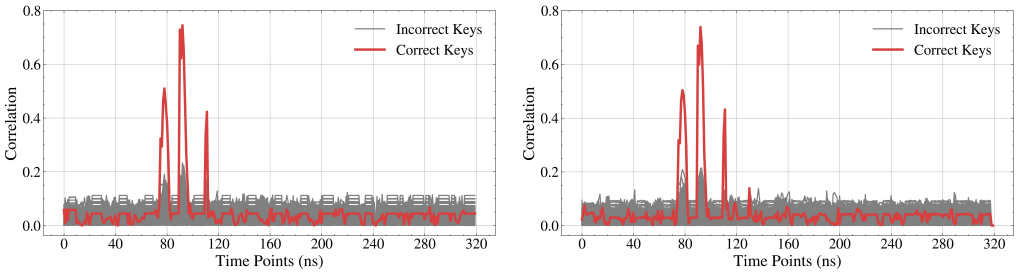
Figure 18 reports the comparison results under visual metrics. On average, AIPS achieves a DTW of 0.0665, a PSD of 6.7620×10^{-14}, and a CS of 0.9999, capturing instruction-aligned transient power structure at the full-core level.

Guided by activity-aligned transients observed during AES instruction execution, we target architectural registers associated with S-Box outputs and apply an HW leakage model. Using both AIPS-generated traces and EDA traces, the same 11 key bytes are



(a) DTW results on evaluation dataset. (b) PSD results on evaluation dataset.

Figure 18: Trace comparison of RISC-V in the time and frequency domains.



(a) CPA results on EDA traces. (b) CPA results on AIPS traces.

Figure 19: CPA results of RISC-V on evaluation dataset.

successfully recovered on the training and evaluation datasets. Figure 19 compares the attack results for the third key byte, showing close agreement between AIPS and EDA despite microarchitectural noise and concurrent non-cryptographic activity.

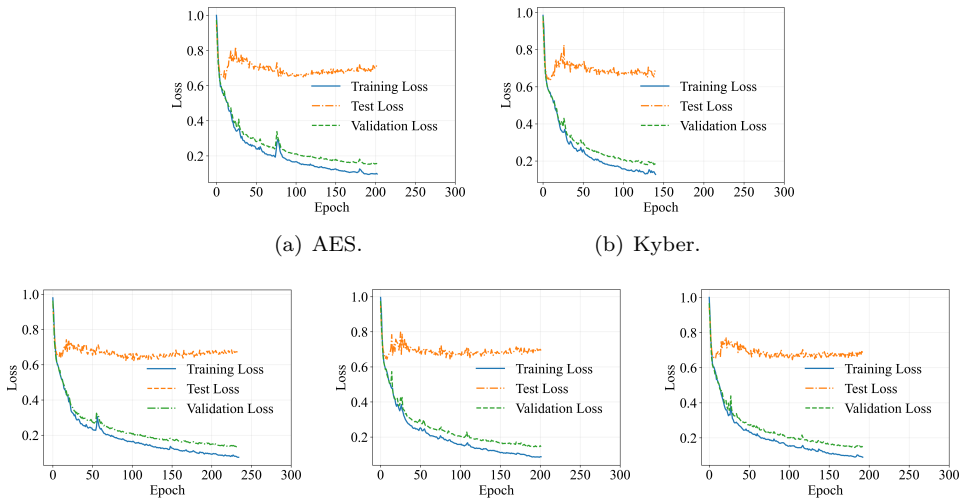
7 Training Performance and Minimum Budget of AIPS

In this section, we analyze both the training performance and the minimum training budget of AIPS to assess its stability and generalization behavior under constrained training conditions.

We first examine the training behavior of AIPS by reporting the training, test, and validation loss curves across all target implementations (in Figure 20). For each target, the training loss decreases steadily and converges within a limited number of epochs. The test curve is computed from only 100 traces and is not used in any of our reported attack results; its higher, slightly increasing loss reflects the high variance one expects from such a small subset and mild overfitting during many training epochs. In contrast, on the fully independent, substantially larger validation set, the training and validation losses closely track each other and flatten to similar values, indicating that the model generalizes well.

Building on the above training behavior, we analyze the impact of training dataset size on AIPS. We consider AES, Kyber, AES_mask_o, and RISC-V, as all have been successfully attacked using CPA in previous experiments. For each target, AIPS is trained with 125, 250, 500, and 1k traces, and used to generate traces for evaluation: 1k for AES, Kyber, and RISC-V, and 20k for AES_mask_o.

The results are summarized in Table 8, which reports both visual and side-channel evaluation metrics. The visual fidelity of traces predicted by AIPS trained on different datasets remains high, but it cannot be guaranteed that these traces can be used for side-channel evaluations. As the training dataset size increases, similarity metrics improve rapidly, but the marginal gains diminish once the training budget exceeds 500 traces. For AES and RISC-V, even models trained with only 125 traces are already sufficient to enable successful CPA. For Kyber and AES_mask_o, the minimum training budget required



(a) AES. (b) Kyber.

(c) AES_mask_o. (d) AES_mask_d. (e) RISC-V.

Figure 20: Loss curves of AIPS.

for a successful attack is 500 traces. Importantly, although 500 EDA traces alone are far from sufficient to directly launch a successful CPA attack, an AIPS model trained on this limited dataset can still learn meaningful leakage patterns and, by generating larger volumes of traces, enable reliable key recovery.

8 Efficiency Performance of AIPS

The efficiency of power trace generation is critical for large-scale design and security evaluation. To quantitatively assess this aspect, we develop an analytical framework that compares the traditional EDA workflow with our proposed AIPS methodology. Our evaluation aims to answer the following questions:

- How does AIPS perform in terms of runtime efficiency on various experimental circuits compared to the EDA method?
- Can AIPS maintain its efficiency advantage as the simulation trace length increases?

8.1 Efficiency Performance Across Different Designs

In our efficiency analysis framework, assuming that the security evaluation of a circuit requires simulating N traces, with $N > 1\text{k}$, the simulation time of the EDA tool becomes:

$$T_{\text{EDA}} = \frac{N}{1000} \times E. \tag{9}$$

Here, E is the EDA simulation time per 1k traces. Similarly, the data preparation and generation time of AIPS can be described by Equation 10.

$$T_{\text{AIPS}} = (E + L + V + D) + \frac{N - 1000}{1000} \times (P + V), \tag{10}$$

where L is the time to parse the lib file, V is the time to parse 1k VCD files, D is the time to train the diffusion model, and P is the time to predict 1k traces.

Table 9 summarizes the parameter values for the above experiments. Since all circuits are implemented on the same technology node and use the same fixed training configuration, the parameters L, D, and P are nearly identical across circuits. However, as the circuit scale increases, the EDA flow encounters significant bottlenecks, with E growing sharply

Table 8: Quality of traces generated by AIPS under different training dataset sizes.

Circuit	Metric	Training dataset size (traces)			
		125	250	500	1000
AES	DTW	0.0897	0.0857	0.0770	0.0744
	PSD	4.9119×10^{-14}	4.1300×10^{-14}	3.0448×10^{-14}	2.5878×10^{-14}
	CS	0.9981	0.9986	0.9993	0.9995
	CPA (1st)	Success	Success	Success	Success
Kyber	DTW	0.1358	0.1124	0.1061	0.0850
	PSD	1.1374×10^{-12}	9.1089×10^{-13}	7.7037×10^{-13}	6.2314×10^{-13}
	CS	0.9999	0.9999	0.9999	0.9999
	CPA (1st)	Failure	Failure	Success	Success
RISC-V	DTW	0.0944	0.0834	0.0749	0.0665
	PSD	9.4246×10^{-14}	8.4674×10^{-14}	6.9963×10^{-14}	6.7620×10^{-14}
	CS	0.9998	0.9999	0.9999	0.9999
	CPA (1st)	Success	Success	Success	Success
AES_mask_o	DTW	0.0439	0.0401	0.0331	0.0322
	PSD	1.0528×10^{-13}	9.4879×10^{-14}	6.9450×10^{-14}	6.3257×10^{-14}
	CS	0.9999	0.9999	0.9999	0.9999
	CPA (2nd)	Failure	Failure	Success	Success

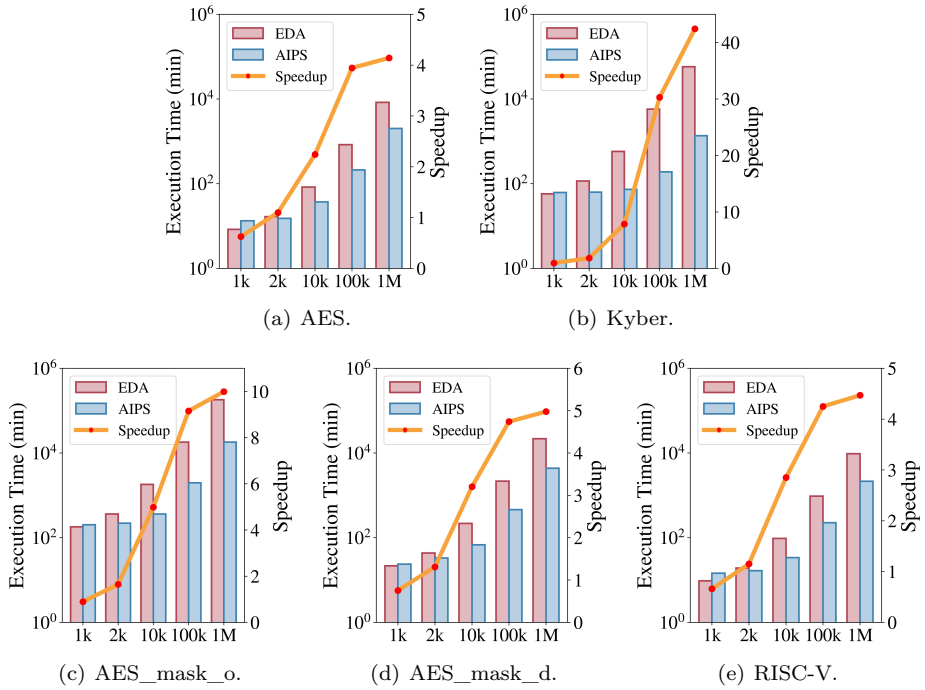
Table 9: Timing parameters for power trace generation, in seconds.

Design	E	L	V	D	P
AES	500	2	120	168	0.05
Kyber	3,467	2	78	177	0.05
AES_mask_o	10,852	2	1,075	170	0.05
AES_mask_d	1,302	2	260	163	0.05
RISC-V	576	2	128	166	0.05

with circuit size. Similarly, V also increases with circuit scale. For Kyber, we only recorded the VCD waveforms for the BU module, resulting in a smaller V.

Figure 21 visualizes the time overhead for generating traces across the five experimental circuits. The x-axis represents the number of traces generated, while the left y-axis records the simulation time (T_{EDA} and T_{AIPS}) in minutes. The right y-axis indicates the speedup ($T_{\text{EDA}}/T_{\text{AIPS}}$), illustrating the efficiency improvement of AIPS. We report speedups under end-to-end deployment (i.e., including L, V, D, and P). The results demonstrate that AIPS starts to show efficiency advantages after 2k traces. As the number of traces and the circuit scale increase, this efficiency advantage becomes more pronounced. For 1M traces, AIPS achieves 4.14× (AES), 42.44× (Kyber), 9.99× (AES_mask_o), 4.98× (AES_mask_d) and 4.47× (RISC-V) faster generation compared to EDA tools. Under equivalent time constraints (the EDA flow generates 1M traces), AIPS generates 4.15×, 44.37×, 10.09×, 5.00× and 4.49× more traces for AES, Kyber, AES_mask_o, AES_mask_d and RISC-V, respectively. This throughput advantage enables more comprehensive security analysis within practical time budgets.

Compared to prior AI-based approaches [ASR+23], AIPS reduces the time for generating 10k AES traces from 30 s to 0.5 s (60× improvement). In a typical use case where only model inference is considered [ZRZ+19] (i.e., considering only P and excluding parsing and training), the speedup reaches more than $10,000\times$ versus EDA tools (i.e., only E).



(a) AES. (b) Kyber.

(c) AES_mask_o. (d) AES_mask_d. (e) RISC-V.

Figure 21: Efficiency comparison of the AIPS and EDA flow.

8.2 Scalability Analysis with Increasing Trace Length

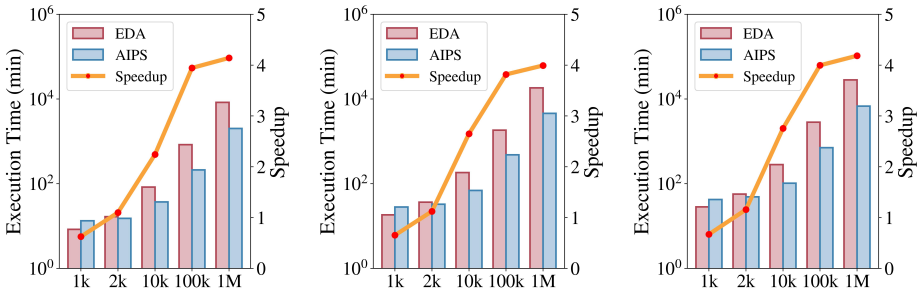
To evaluate the scalability and stability of AIPS under varying trace lengths, we run experiments using AES circuits at three simulation durations: 160 ns, 320 ns, and 480 ns. Table 10 summarizes the parameters involved in generating 1k traces in each case. As the trace length increases, the simulation time for EDA and AIPS increases proportionally. Notably, the parameters E, V, D, and P exhibit approximately linear relationships with increasing trace length, suggesting that computational cost is predictable. Therefore, we compare their performance across different data sizes using Equation 9 and 10.

Table 10: Timing parameters for power trace generation, in seconds.

Design (trace length)	E	L	V	D	P
AES (160 ns)	500	2	120	168	0.05
AES (320 ns)	1,100	2	274	309	0.06
AES (480 ns)	1,700	2	404	428	0.08

Figure 22 compares simulation time and speedup of AIPS against the EDA flow across trace lengths. For 160 ns, 320 ns, and 480 ns, both runtimes grow approximately linearly with trace length (Table 10), and the AIPS/EDA speedup remains similar (Figure 22), indicating that AIPS retains its relative efficiency advantage as the trace window increases.

Although our evaluation focuses on leakage-relevant time windows, which is common in pre-silicon SCA due to the high cost of obtaining very long PTPX traces, AIPS does not inherently prevent full-trace generation. The same trained model can be applied to successive, possibly overlapping windows (e.g., round by round), and the resulting segments can be concatenated to form longer traces or complete executions. Similar strategies are common in diffusion-based audio waveform synthesis [KPH+20, CZZ+20].



(a) AES (160 ns). (b) AES (320 ns). (c) AES (480 ns)

Figure 22: Efficiency comparison of the AIPS and EDA flow.

9 Conclusion

We introduced AIPS, a diffusion-driven pre-silicon power-trace generation framework that generates gate-level transient power traces for side-channel evaluation. Across AES, Kyber, masked AES, and a RISC-V core, AIPS reproduces the same outcomes as PTPX under the evaluated CPA and TVLA tests, while providing significant speedups for large trace volumes. AIPS is open source [GMZ+26].

At present, migrating AIPS to new circuits or technology nodes requires retraining on the target using VCD activity, cell features, and reference power traces. This aligns with common practice in AI-driven circuit modeling, where cross-node deployment is typically achieved via transfer learning or light retraining on a small number of samples, rather than expecting a single model to generalize across processes without adaptation. A natural direction for future work is to investigate lightweight fine-tuning to reduce this adaptation cost and improve portability across architectures and technology nodes. Furthermore, to support finer-grained security diagnosis, another promising direction is a top-down hierarchical localization workflow. In such a workflow, AIPS would first identify risks at the full-design or module level, then progressively narrow down to suspicious submodules and candidate leaking regions or cells by integrating finer-grained side-channel evaluation techniques. Together, these directions could further improve the portability, scalability, and practical value of AIPS for secure hardware design.

Acknowledgments

We thank the anonymous reviewers and the shepherd for their valuable feedback. This research was supported by the National Key R&D Program of China (Grant 2023YFB4402800). Part of this work was completed during Ya Gao's stay at EPFL; this visit was supported in part by the China Scholarship Council.

References

[ABPP21] Vipul Arora, Ileana Buhan, Guilherme Perin, and Stjepan Picek. A tale of two boards: On the influence of microarchitecture on side-channel leakage. In *International Conference on Smart Card Research and Advanced Applications (CARDIS)*, pages 80–96, 2021.

[ASR+23] Abdullah Aljuffri, Mudit Saxena, Cezar Reinbrecht, Said Hamdioui, and Mottaqiallah Taouil. A pre-silicon power leakage assessment based on generative adversarial networks. In *Euromicro Conference on Digital System Design (DSD)*, pages 87–94, 2023.

[ASS24] Jean-Eudes Ayilo, Mostafa Sadeghi, and Romain Serizel. Diffusion-based speech enhancement with a weighted generative-supervised learning loss. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12506–12510, 2024.

[BBJP19] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *USENIX Security Symposium (USENIX Security)*, pages 515–532, 2019.

[BBYS22] Ileana Buhan, Lejla Batina, Yuval Yarom, and Patrick Schaumont. SoK: Design tools for side-channel-aware implementations. In *Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 756–770, 2022.

[Cav24] Brandon Cavins. riscv-cpu: A 32-bit RISC-V CPU core. https://github.com/bzeeno/riscv-cpu, 2024. Accessed: 2025-01.

[CZZ+20] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, pages 1–15, 2020.

[DN21] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:8780–8794, 2021.

[DSM22] Siemen Dhooghe, Aein Rezaei Shahmirzadi, and Amir Moradi. Second-order low-randomness $d + 1$ hardware sharing of the AES. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 815–828, 2022.

[FNS24] Philipp Fengler, Sani Nassif, and Ulf Schlichtmann. Toward early stage dynamic power estimation: Exploring alternative machine learning methods and simulation schemes. In *International Symposium on Quality Electronic Design (ISQED)*, pages 1–8, 2024.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance validation. In *NIST Non-Invasive Attack Testing Workshop*, volume 7, pages 115–136, 2011.

[GMZ+26] Ya Gao, Haocheng Ma, Tanchen Zhang, Jiaji He, Yiqiang Zhao, Mirjana Stojilović, and Yier Jin. AIPS: AI-based power simulation for pre-silicon side-channel security evaluation: Artifacts. https://doi.org/10.5281/zenodo.18158824, https://github.com/jinyier/AIPS, 2026. Accessed: 2026-01.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.

[HPN+19] Miao He, Jungmin Park, Adib Nahiyan, Apostol Vassilev, Yier Jin, and Mark Tehranipoor. RTL-PSC: Automated power side-channel leakage assessment at register-transfer level. In *VLSI Test Symposium (VTS)*, pages 1–6, 2019.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, pages 1–15, 2015.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *International Cryptology Conference (CRYPTO)*, pages 388–397, 1999.

[KLS22] Pantea Kiaei, Zhenyuan Liu, and Patrick Schaumont. Leverage the average: Averaged sampling in pre-silicon side-channel leakage assessment. In *Great Lakes Symposium on VLSI (GLSVLSI)*, pages 3–8, 2022.

[KPH+20] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, pages 1–17, 2020.

[KPP24] Sengim Karayalcin, Guilherme Perin, and Stjepan Picek. Diffuse some noise: Diffusion models for measurement noise removal in side-channel analysis. Cryptology ePrint Archive, Paper 2024/966, 2024.

[LDR+22] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, 2022.

[LLF+25] Zeng Lu, Yan Longde, Wan Fei, Chen Aidong, Yang Ning, Li Xiang, Zhang Jiancheng, Zhang Yanlong, Wang Shuo, and Zhou Jing. Scarefusion: Side channel analysis data restoration with diffusion model. *Microelectronics Journal*, 156:106546, 2025.

[LPH16] Jordane Lorandel, Jean-Christophe Prévotet, and Maryline Hélard. Efficient modelling of FPGA-based IP blocks using neural networks. In *International Symposium on Wireless Communication Systems (ISWCS)*, pages 571–575, 2016.

[LS24] Zejia Lyu and Jizhong Shen. An efficient algorithm for estimating gate-level power consumption in large-scale integrated circuits. *Microelectronics Journal*, 146:106143, 2024.

[LZX24] Yao Lu, Qijun Zhang, and Zhiyao Xie. Unleashing flexibility of ML-based power estimators through efficient development strategies. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6, 2024.

[MNBV23] Kazuki Monta, Makoto Nagata, Josep Balasch, and Ingrid Verbauwhede. On the unpredictability of SPICE simulations for side-channel leakage verification of masked cryptographic circuits. In *Design Automation Conference (DAC)*, pages 1–6, 2023.

[MPG+22] Haocheng Ma, Shijian Pan, Ya Gao, Jiaji He, Yiqiang Zhao, and Yier Jin. Vulnerable PQC against side channel analysis-a case study on kyber. In *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6, 2022.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 157–171, 2005.

[Nes18] Sondre Rennan Nesset. RTL power estimation flow and its use in power optimization. Master's thesis, Norwegian University of Science and Technology, 2018.

[NSP+20] Yehya Nasser, Carlo Sau, Jean-Christophe Prévotet, Tiziana Fanni, Francesca Palumbo, Maryline Hélard, and Luigi Raffo. Neupow: A CAD methodology for high-level power estimation based on machine learning. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 25(5):1–29, 2020.

[OMHT06] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical second-order DPA attacks for masked smart card implementations of block ciphers. In *Cryptographers' Track at the RSA Conference (CT-RSA)*, pages 192–207, 2006.

[OMPR05] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A side-channel analysis resistant description of the AES S-box. In *International Workshop on Fast Software Encryption (FSE)*, pages 413–423, 2005.

[PGA+23] Kostas Papagiannopoulos, Ognjen Glamočanin, Melissa Azouaoui, Dorian Ros, Francesco Regazzoni, and Mirjana Stojilović. The side-channel metrics cheat sheet. *ACM Computing Surveys*, 55(10):1–38, 2023.

[PPFT22] Nitin Pundir, Jungmin Park, Farimah Farahmandi, and Mark Tehranipoor. Power side-channel leakage assessment framework at register-transfer level. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(9):1207–1218, 2022.

[PWP22] Guilherme Perin, Lichao Wu, and Stjepan Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, 2022(4):828–861, 2022.

[RDTA23] MB Rakesh, Pabitra Das, Anant Terkar, and Amit Acharyya. GRASPE: Accurate post-synthesis power estimation from RTL using graph representation learning. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2023.

[RPG+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning (ICML)*, pages 8821–8831, 2021.

[RRD+23] Gokulnath Rajendran, Prasanna Ravi, Jan-Pieter D'anvers, Shivam Bhasin, and Anupam Chattopadhyay. Pushing the limits of generic side-channel attacks on LWE-based KEMs: Parallel PC-oracle attacks on Kyber and beyond. *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, pages 418–446, 2023.

[Saa20] Markku-Juhani O Saarinen. A lightweight ISA extension for AES and SM4. *arXiv preprint arXiv:2002.07041*, pages 1–4, 2020.

[ŠBY+20] Danilo Šijačić, Josep Balasch, Bohan Yang, Santosh Ghosh, and Ingrid Verbauwhede. Towards efficient and automated side-channel evaluations at design time. *Journal of Cryptographic Engineering*, 10:305–319, 2020.

[SC07] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[SCC+22] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH Conference (SIGGRAPH)*, pages 1–10, 2022.

[SCS+22] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:36479–36494, 2022.

[SDME21] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.

[SGS23] David Spielmann, Ognjen Glamočanin, and Mirjana Stojilović. RDS: FPGA routing delay sensors for effective remote power analysis attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, pages 543–567, 2023.

[SGZ+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in Neural Information Processing Systems (NeurIPS)*, 29:1–9, 2016.

[SM+05] Petre Stoica, Randolph L Moses, et al. *Spectral analysis of signals*, volume 452. Prentice Hall, Upper Saddle River, New Jersey, USA, 2005.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology: A clear roadmap for side-channel evaluations. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 495–513, 2015.

[SMRM19] Rajat Sadhukhan, Paulson Mathew, Debapriya Basu Roy, and Debdeep Mukhopadhyay. Count your toggles: A new leakage model for pre-silicon power analysis of crypto designs. *Journal of Electronic Testing*, 35:605–619, 2019.

[SS23] Dillibabu Shanmugam and Patrick Schaumont. Improving side-channel leakage assessment using pre-silicon leakage models. In *Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*, pages 105–124, 2023.

[TG06] Stefan Tillich and Johann Großschädl. Instruction set extensions for efficient AES implementation on 32-bit processors. In *International workshop on cryptographic hardware and embedded systems*, pages 270–284, 2006.

[THH+05] Kris Tiri, David Hwang, Alireza Hodjat, Bo-Cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. Prototype IC with WDDL and differential routing–DPA resistance assessment. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 354–365, 2005.

[UJP24] Brian Udugama, Darshana Jayasinghe, and Sri Parameswaran. Sensors for remote power attacks: New developments and challenges. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 333–340, 2024.

[Viv24] Sanchayan Vivekananthan. Comparative analysis of generative models: Enhancing image synthesis with vaes, gans, and stable diffusion. *arXiv preprint arXiv:2408.08751*, 2024.

[VWGV+23] Jasper Van Woudenberg, Peter Grossmann, Avinash L Varna, Joseph Friel, Daniel Dinu, Ronnie Lindsay, and Steve J Brown. Pre-silicon side channel and fault analysis. In *Design Automation Conference (DAC)*, pages 1–4, 2023.

[YJ24] Trevor Yap and Dirmanto Jap. Creating from noise: Trace generation using diffusion models for side-channel attack. In *International Conference on Applied Cryptography and Network Security (ACNS)*, pages 102–120, 2024.

[YMZ+15] Jianlei Yang, Liwei Ma, Kang Zhao, Yici Cai, and Tin-Fook Ngai. Early stage real-time soc power estimation using rtl instrumentation. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 779–784, 2015.

[ZLYW23] Shiduo Zhang, Xiuhan Lin, Yang Yu, and Weijia Wang. Improved power analysis attacks on falcon. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 565–595, 2023.

[ZPM+23] Yiqiang Zhao, Shijian Pan, Haocheng Ma, Ya Gao, Xintong Song, Jiaji He, and Yier Jin. Side channel security oriented evaluation and protection on hardware implementations of kyber. *IEEE Transactions on Circuits and Systems I: Regular Papers*, pages 5025–5035, 2023.

[ZRK20] Yanqing Zhang, Haoxing Ren, and Brucek Khailany. GRANNITE: Graph neural network inference for transferable power estimation. In *Design Automation Conference (DAC)*, pages 1–6, 2020.

[ZRSK22] Yanqing Zhang, Haoxing Ren, Akshay Sridharan, and Brucek Khailany. GATSPI: GPU accelerated gate-level simulation for power improvement. In *Design Automation Conference (DAC)*, pages 1231–1236, 2022.

[ZRZ+19] Yuan Zhou, Haoxing Ren, Yanqing Zhang, Ben Keller, Brucek Khailany, and Zhiru Zhang. PRIMAL: Power inference using machine learning. In *Design Automation Conference (DAC)*, pages 1–6, 2019.