Early Detection of Hardware Trojans Using Neural Controlled Differential Equations and Analysis of Power Traces

Hasala Senevirathne
Dept. of Computer Eng. & Computer Sci.
California State University, Long Beach
Long Beach, CA, United States
hasala.senevirathne01@student.csulb.edu

Rahul Vishwakarma
Dept. of Computer Eng. & Computer Sci.
California State University, Long Beach
Long Beach, CA, United States
rahuldeo.vishwakarma01@student.csulb.edu

Amin Rezaei
Dept. of Computer Eng. & Computer Sci.
California State University, Long Beach
Long Beach, CA, United States
amin.rezaei@csulb.edu

***Abstract*—Evolving Hardware Trojans pose a serious threat to modern digital systems by evading traditional detection through stealthy, adaptive behavior. Even recent methods that leverage advances in machine learning can only detect them after activation, leaving a critical window for potential security breaches. To address this gap, we propose a novel approach for hardware Trojan detection and prediction using Neural Controlled Differential Equations (NCDEs) and analysis of power traces. Our method leverages an NCDE model trained exclusively on Trojan-free data to learn nominal power behavior, combined with a Linear Discriminant Analysis (LDA) classifier calibrated on labeled data, to distinguish between three scenarios: no Trojan, dormant Trojan, and active Trojan. Our method uses a sliding window to process side-channel measurements, enabling detection of subtle power consumption deviations that indicate Trojan presence, even when dormant. Experimental results demonstrate that the proposed NCDE-based method achieves superior accuracy compared to traditional machine learning approaches, with the additional advantage of handling dormant Trojans above a sensitivity threshold. We validate our approach on standard hardware Trojan benchmarks, showing robust detection and prediction performance.**

***Index Terms*—Hardware Trojan Detection, Neural Controlled Differential Equations, Power Side-Channel Analysis**

I. Introduction

Evolving Hardware Trojans (HTs) threaten electronic systems by altering Integrated Circuits (ICs), leading to unauthorized access, data leakage, or system failure [1]. Their stealthy and adaptive nature allows HTs to evade traditional detection methods, making them a key concern in hardware security research. Traditional static detection methods, such as static analysis [2], [3] and logic testing [4], struggle with scalability and are impractical for identifying Trojans with rare activation triggers. As ICs grow in size and complexity, exhaustive testing becomes too complex. Dynamic detection methods based on neural networks have been explored [5], but models like Recurrent Neural Networks (RNNs) [6] require significant computational resources and may not capture the irregular, high-dimensional side-channel signals, such as power consumption and electromagnetic emissions, needed to detect Trojans at run-time.

A particularly challenging aspect of HT detection is identifying not only when a Trojan is actively triggered but also when it is present yet dormant in the circuit. This capability is essential for proactive security measures, as it allows for the prediction of malicious hardware before it can cause harm. Traditional detection methods often focus solely on identifying active Trojans, missing the opportunity to detect dormant threats [7]–[24].

Neural Controlled Differential Equations (NCDEs) have recently gained attention for their ability to model continuous-time dynamics in irregular time-series data [25]. NCDEs extend neural networks into continuous time, effectively capturing intricate temporal dependencies and accommodating irregularly sampled observations. This makes them particularly well-suited for processing side-channel signals that are inherently irregular and high-dimensional. We believe that by modeling the temporal evolution of these signals, NCDEs have the potential to identify subtle deviations from normal behavior, indicating the presence of dormant Trojans even when they are not actively triggered.

In this paper, we investigate the potential of using NCDEs combined with a sliding window mechanism to analyze power consumption traces, enabling not only the detection of HTs upon activation but also the prediction of their presence while they remain dormant. Our method leverages an NCDE model trained on Trojan-free data, combined with an LDA classifier calibrated on labeled traces, to distinguish between three scenarios: (1) No Trojan presented, (2) Trojan presented but dormant, and (3) Trojan presented and active. This three-way classification provides a more nuanced view of system security than traditional binary classification approaches. Our main contributions are as follows:

- Proposing a novel three-way classification approach for detection and prediction of HTs, leveraging NCDEs to effectively model power side-channel signals;
- Exploring a threshold at which the model can reliably catch the presence of HTs, even while they remain dormant;
- Implementing an efficient sliding window approach for processing power trace data, enabling systematic Trojan detection and state classification.

The rest of the paper is organized as follows: Section II reviews related works on HT detection and time series. Section III covers the preliminaries on HTs, side-channel analysis, and NCDEs. Section IV outlines our methodology for early Trojan detection using NCDEs. Section V presents the experimental setup, results, and analysis. Finally, Section VI concludes the paper and discusses future works.

II. Related Works

A. Hardware Trojan Detection

HT detection approaches are broadly categorized into pre-silicon [26] and post-silicon [27] methods. Pre-silicon techniques focus on design-time verification, while post-silicon methods involve physical inspection and side-channel analysis [28]. A comprehensive survey [29] underscores the significant challenges in detecting dormant hardware Trojans, a key focus of this work.

Side-channel analysis has proven effective. Path delay fingerprinting [30] identifies timing anomalies but requires extensive characterization. Power consumption analysis [31] compares measurements with golden models but struggles with process variations. ML approaches include Multi-Layer Neural Networks (MLNNs) [32] and Long Short-Term Memories (LSTMs) [33] for analyzing power

979-8-3315-5193-3/26/$31.00

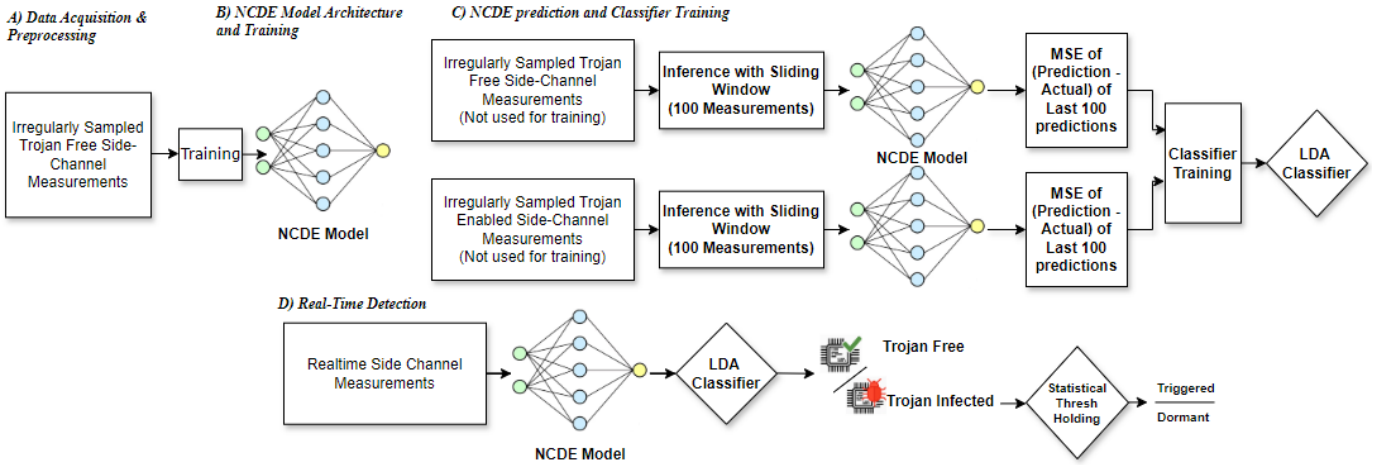


Fig. 1: **HOODOO**: NCDE-based Hardware Trojan Detection and Prediction Framework

traces, though they require extensive training data and may not capture continuous-time dynamics. A brain-inspired model known as Hierarchical Temporal Memory (HTM) is proposed for HT detection and is designed to be resilient to natural variations in side-channel measurements [34]. Recently, the application of Large Language Models (LLMs) to hardware Trojan (HT) detection has also been explored, both with and without the inclusion of contextual supplementary information [35]. However, these efforts are still in the early stages, and further research is needed to evaluate the effectiveness of LLMs in detecting zero-day HTs.

B. Time Series Analysis

Neural Ordinary Differential Equations (Neural ODEs) [36] introduced continuous-time generalization of residual networks, representing hidden state dynamics as ODEs for flexible continuous-time modeling. Building on Neural ODEs, NCDEs [25] were proposed for irregularly sampled time series, parameterizing the ODE vector field using neural networks with control paths from observed data. The NCDE framework has been extended for noisy/partially observed series with improved gradient computation and stability [37], crucial for noisy side-channel analysis. GRU-ODE-Bayes [38] combines gated recurrent units with neural ODEs for variable sampling rates in monitoring scenarios. While these approaches show promise across domains, their hardware security and HT detection applications remain unexplored. Our work bridges this gap by adapting NCDEs to side-channel-based HT detection problem.

Our work leverages the continuous-time modeling capabilities of NCDEs within a sliding window framework, enabling high detection (when active) and prediction (when dormant) accuracy of HTs, without requiring hardware modifications.

III. Preliminaries

A. Hardware Trojans

HTs are malicious IC modifications that modify functionality, leak information, or cause system failure [39]. They consist of a trigger and a payload, with the trigger activating the Trojan under specific hard-to-detect conditions and the payload executing the malicious function. Our approach identifies three scenarios: (1) **No Trojan**: the circuit is clean; (2) **Dormant Trojan**: a Trojan is physically present but its trigger condition has not been met, its presence may still create a measurable, albeit subtle, side-channel footprint; and (3) **Active Trojan**: the Trojan is triggered and executes its payload, producing a pronounced deviation in power consumption.

B. Side-Channel Analysis

Side-channel analysis leverages physical characteristics like power consumption, timing, or electromagnetic emissions to infer circuit operations [40]. Power analysis is widely used for HT detection, measuring IC power consumption and comparing it with expected profiles [29]. Deviations may indicate Trojan presence. Its advantages include non-invasiveness, high sensitivity to behavioral changes, and broad applicability across circuit types.

However, process variations can mask Trojan-induced changes, environmental noise reduces detection sensitivity, and modern ICs produce intricate signatures where irregular sampling may miss critical events. NCDEs address these challenges by effectively modeling continuous-time power signal dynamics and handling irregularly sampled or noisy data.

C. Neural Controlled Differential Equations

Unlike traditional RNNs that operate in discrete time steps, NCDEs model the evolution of hidden states as a continuous-time process controlled by the input data. Mathematically, an NCDE is defined as:

$$h(t) = h(t_0) + \int_{t_0}^{t} f_\theta(h(s))\, dX(s), \tag{1}$$

where $h(t)$ is the hidden state at time t, f_θ is a neural network with parameters θ governing the hidden state dynamics, $X(t)$ is a continuous interpolation of the input data, and $dX(s)$ represents the differential of the control path. In practice, input data points are interpolated to create $X(t)$ using cubic splines or Hermite cubic interpolation [41], and the equation is solved numerically using methods like Runge-Kutta. NCDEs naturally handle irregular sampling, capture intricate temporal dynamics through continuous-time modeling, and offer architectural flexibility—properties well-suited for HT detection using power measurements.

IV. Trojan Detection and Prediction

In this section, we present **HOODOO**, a **H**ardware Tr**O**jan detecti**O**n and pre**D**icti**O**n framew**O**rk using NCDEs shown in Fig. 1.

Algorithm 1 NCDE Function

function CDEFUNC(h, $input\_dim$, $hidden\_dim$)
    $batch\_size \leftarrow$ size of first dimension of h
    $x \leftarrow \mathbf{ones}(batch\_size, input\_dim)$
    $xz \leftarrow$ concatenate(x, h)
    $hidden \leftarrow$ Linear$(xz, hidden_dim \times 2)$
    $hidden \leftarrow$ GELU$(hidden)$
    $z \leftarrow$ Linear$(hidden, input_dim \times hidden_dim)$
    return reshape(z, [$batch\_size$, $hidden\_dim$, $input\_dim$])
end function

The central idea is to train an NCDE model on power trace data obtained from Trojan-free hardware to learn its nominal behavior. We then use this model to identify anomalies in new measurements that may suggest the presence of a Trojan, distinguishing between dormant and active states. **HOODOO** consists of the following components: **A** **Data Acquisition and Preprocessing**, which involves collecting power consumption traces from the device under test, followed by normalization and sliding window segmentation of these traces; **B** **NCDE Model Architecture and Training**, a time-series model trained and optimized on Trojan-free data; **C** **NCDE Prediction and Classifier Training**, which employs the trained NCDE model and labeled power trace data to train the classifier. **D** **Detection and Classification**, which categorizes the device as non-Trojan, dormant Trojan, or active Trojan states, by feeding power trace data to the NCDE model and then to the classifier.

A. Data Acquisition and Preprocessing

We employ a publicly accessible online dataset [42] with high-resolution power consumption traces from hardware platforms in different operational states. To process power trace data, we use a sliding window technique with a buffer of W samples. The window advances by removing old and adding new measurements. For each window, we apply three pre-processing steps:

1) **Normalization**: Subtract mean and divide by standard deviation from training data: $x_{\text{normalized}} = \frac{x - \mu_{\text{train}}}{\sigma_{\text{train}}}$, where μ_{train} and σ_{train} are training dataset statistics. This ensures zero mean and unit variance for stable training.
2) **Time Step Assignment**: Assign equidistant time steps in $[0, 1]$: $t_i = \frac{i}{W-1}$ for $i = 0, 1, \ldots, W-1$.
3) **Interpolation Coefficient Computation**: Compute cubic spline or Hermite cubic interpolation coefficients to represent the data as a continuous path.

The normalized window and interpolation coefficients are fed to the NCDE model for processing.

B. NCDE Model Architecture and Training

Our NCDE model architecture consists of three main components:

1) **Initial Mapping**: A linear layer that maps the initial data point in the window to the initial hidden state: $h(t_0) = \text{Linear}(X(t_0))$.
2) **CDE Function**: A neural network that defines the dynamics of the hidden state: $f_\theta(h(t)) = \text{NeuralNetwork}(h(t))$.
3) **Readout Layer**: A linear layer that maps the final hidden state to the predicted output: $\hat{y} = \text{Readout}(h(t))$.

Algorithm 1 parametrizes the CDE vector field: it concatenates a bias vector of ones with the hidden state h, processes this through a two-layer MLP with GELU activation, and reshapes the output for CDE integration.

Algorithm 2 NCDE Model Forward Pass

function NCDEFORWARD($coeffs$, $input\_dim$, $hidden\_dim$, $output\_dim$)
    # Create continuous path
    $X \leftarrow$ CubicSpline$(coeffs)$
    # Initial point
    $X_0 \leftarrow X$.evaluate$(X.interval[0])$
    # Initial hidden state
    $z_0 \leftarrow$ Linear$(X_0, hidden_dim)$
    # Solve differential equation with numerical method
    $h_T \leftarrow$ CDEint$(X, z_0,$ CDEFunc$, X.grid_points,$
        $method =$ 'rk4')
    # Apply readout
    $output \leftarrow$ Linear$(h_T[:, -1, :], output_dim)$
    return $output$
end function

Algorithm 3 NCDE-based Trojan Detection

function DETECTTROJAN($model$, $power\_trace$, W, b_{LDA}, $T_{triggered}$)
    $errors \leftarrow []$
    for $i \leftarrow 0$ to len$(power_trace) - W - 1$ **do**
        $window \leftarrow power\_trace[i : i + W]$
        $norm\_window \leftarrow$ Normalize$(window)$
        $coeffs \leftarrow$ ComputeCoefficients$(norm_window)$
        # NCDE predicts the next sample
        $\hat{y} \leftarrow model(coeffs)$
        # Actual next sample
        $y \leftarrow power\_trace[i + W]$
        # Squared prediction error
        $errors$.append$((\hat{y} - y)^2)$
    end for
    $MSE \leftarrow$ mean$(errors)$
    if $MSE \leq b_{LDA}$ **then**
        return "No Trojan Presented"
    else if $MSE \leq T_{triggered}$ **then**
        return "Trojan Presented but Dormant"
    else
        return "Trojan Presented and Active"
    end if
end function

Algorithm 2 implements the forward pass: it creates a continuous path via cubic spline interpolation, initializes the hidden state through a linear mapping, solves the CDE using 4th-order Runge-Kutta, and extracts the final hidden state for prediction through a readout layer.

Algorithm 3 implements the three-state classification: for each sliding window of W samples, it predicts the next power sample via the NCDE model, computes the squared prediction error, and classifies the aggregate MSE using the LDA-derived threshold b_{LDA} and the active Trojan threshold $T_{triggered}$.

We train our NCDE model exclusively on Trojan-free power traces to predict the next power sample given a window of W preceding samples. By learning normal circuit behavior, the model produces higher prediction errors on Trojan-infected circuits, which the LDA classifier leverages for detection. Training optimizes the NCDE parameters to minimize MSE between predicted and actual next power values:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \left(y_i - \hat{y}_i\right)^2, \tag{2}$$

where y_i is the actual next power value and $\hat{y}_i$ is the predicted value.

We also employ several optimization techniques to improve training efficiency:

1) **AdamW optimizer** with weight decay for regularization.
2) **OneCycleLR scheduler** for dynamic learning rate adjustment.
3) **Mixed-precision training** for faster computation on compatible GPUs.
4) **Gradient scaling** to prevent underflow in mixed-precision training.

C. NCDE Prediction and Classifier Training

During inference, for each window of W samples starting at index i, the NCDE model predicts the next value $\hat{y}_{i+W}$, and the squared prediction error is computed:

$$e_i = (\hat{y}_{i+W} - y_{i+W})^2. \tag{3}$$

These errors are aggregated over N windows to obtain the MSE:

$$\text{MSE} = \frac{1}{N} \sum_{i=0}^{N-1} e_i. \tag{4}$$

A higher MSE indicates greater deviation from nominal behavior. We employ Linear Discriminant Analysis (LDA) for classification, where the feature is the MSE value and the classes are Trojan-free (C_0) and Trojan-infected (C_1). While the NCDE model trains on Trojan-free data only, the LDA classifier requires a small labeled calibration set from both classes. LDA maximizes the between-class to within-class variance ratio, determining an optimal boundary b_{LDA}. Classification of a new trace proceeds as:

$$\text{Classification} = \begin{cases} \text{Trojan-free} & \text{if } \text{MSE}_{new} \leq b_{LDA} \\ \text{Trojan Dormant} & \text{if } b_{LDA} < \text{MSE}_{new} \leq T_{triggered} \\ \text{Trojan Active} & \text{if } \text{MSE}_{new} > T_{triggered} \end{cases} \tag{5}$$

where b_{LDA} is the optimal threshold from LDA. The second threshold $T_{triggered}$ distinguishes dormant from active states; however, its calibration requires labeled traces with known trigger timestamps, which are unavailable in the current benchmark. As this work focuses on dormant Trojan detection, the experimental evaluation centers on b_{LDA}, while the dormant-versus-active distinction is validated through noise injection at varying intensity levels.

V. Experimental Results

A. Experimental Setup

We used the publicly available hardware Trojan power side-channel dataset [42], which contains power traces measured from physical hardware using a Sakura-G FPGA board and a Tektronix TDS2022C oscilloscope. The dataset includes TrustHub benchmarks with two base circuits: an AES-128 cryptography core (9707 LUTs) and an RS232 UART serial communication circuit, infected with six Trojan variants: AES-T500, AES-T600, AES-T700, AES-T800, AES-T1600, and RS232-T100. These Trojans range from 0.29% to 4.46% of the base circuit area and implement diverse payloads including denial of service, secret key leakage through leakage current and covert channels, and RF transmission. Power traces were collected under two conditions per benchmark: HT inactive (dormant) and HT activated (triggered), along with Trojan-free baseline traces. Each category contains 10,000 traces of 2,500 samples each. Process variation was addressed in the original dataset by collecting traces from two separate Sakura-G boards.

Power traces were segmented into sliding windows of $W = 50$ samples with a stride of 1 sample. Trojan-free data was split 80/20 for training/validation using fixed-seed random splitting. The LDA classifier was calibrated using MSE values derived from both Trojan-free and Trojan-infected traces. The NCDE model was implemented in PyTorch with configurations in Tables I and II. The experiments ran on a 12-core Intel Xeon Silver 4310 processor with 16GB of DDR4 RAM and an NVIDIA 64GB A16 GPU.

TABLE I: NCDE Model Configuration

Parameter	Value
Input channels	2 (time, power)
Hidden channels	64
Output channels	1 (predicted power)
CDE function	2-layer MLP, GELU, dropout 0.1
Numerical solver	RK4, step size 0.2
Batch size	256 (GPU), 64 (CPU)
Epochs	30

TABLE II: Training Configuration

Parameter	Value
Optimizer	AdamW, LR 1×10^{-3}, WD 1×10^{-4}
LR scheduler	OneCycleLR, max LR 1×10^{-3}
Loss function	Mean Squared Error (MSE)
Mixed precision	Enabled (if GPU supports)

B. Evaluation Methodology

We evaluated our approach using the following methodology:

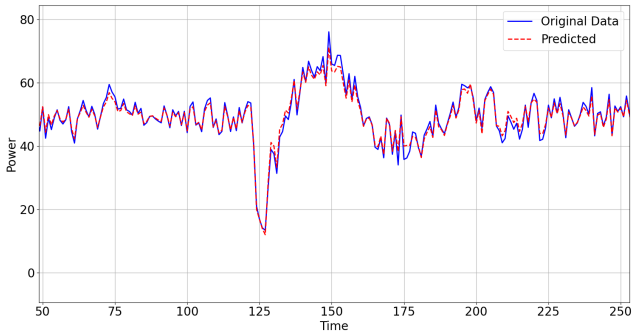
1) **Train-Test Split**: The NCDE model was trained exclusively on Trojan-free power traces. Testing was performed on separate sets for each category (no Trojan, dormant, active).
2) **Error Metrics**: For each sliding window of $W = 50$ samples, the NCDE model predicts the next power sample, and the squared prediction error is computed. The MSE is aggregated across all windows in a trace.
3) **Threshold Determination**: An LDA classifier was trained on MSE values from both Trojan-free and Trojan-infected traces to determine the optimal decision boundary b_{LDA} for separating clean and Trojan-infected states. The dormant-versus-active distinction is evaluated through noise injection at varying intensity levels (1–5% of peak amplitude), as the benchmark dataset does not provide explicit trigger timestamps for calibrating $T_{triggered}$.
4) **Classification**: Each trace is classified based on its MSE: $\text{MSE} \leq b_{LDA} \Rightarrow$ No Trojan; $\text{MSE} > b_{LDA} \Rightarrow$ Trojan Detected (dormant or active, distinguished by deviation magnitude).

C. NCDE Model Performance

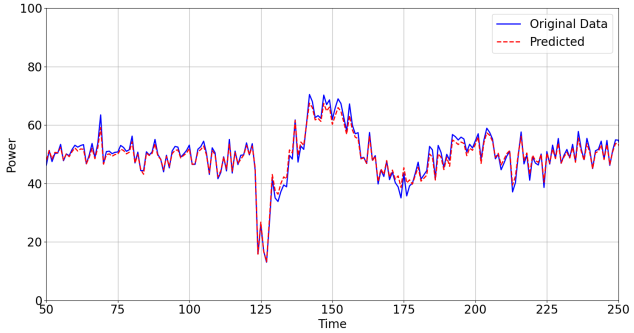
Fig. 2a validates the NCDE model's baseline performance in predicting nominal circuit behavior in the absence of Trojans. Fig. 2b quantifies the prediction residuals observed when dormant Trojans introduce subtle perturbations in power consumption patterns. Fig. 2c exhibits the model's response to active Trojan payloads, contrasting predicted power traces with actual measurements to highlight the detectable deviations. As shown in the figures, the NCDE model demonstrates strong performance in predicting continuous values.

D. HT Detection Performance

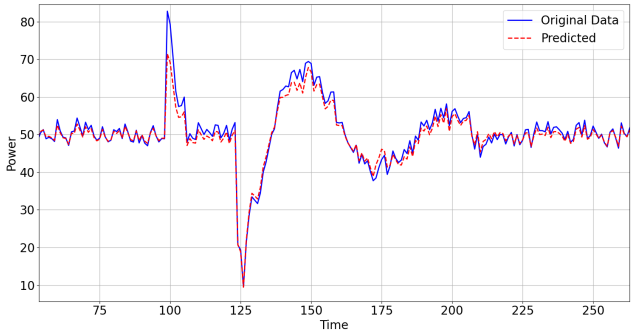
To evaluate detection sensitivity, we injected normally distributed noise at varying levels (1–5% of peak power trace amplitude) into Trojan-free traces, simulating the subtle power deviations that stealthy hardware Trojans introduce to evade detection [43]. This provides a controlled evaluation scenario for assessing the minimum



(a) Trojan Disabled



(b) Trojan Dormant



(c) Trojan Triggered

Fig. 2: NCDE Predictions under Different HT conditions

perturbation level at which the model can reliably distinguish Trojan-affected behavior from nominal operation. As shown in Fig. 3, our NCDE-based method successfully detected Trojans at noise levels $\geq$3% of peak power value. Below this 3% threshold, the model's ability to distinguish dormant Trojans from normal behavior degraded, indicating a critical detection boundary where Trojans inducing variations $<$3% can evade detection, highlighting the need for enhanced sensitivity.

E. Comparison with State-of-the-Art Methods

Table III compares our approach with recent ML-based HT detection methods [32]–[35]. The compared methods use different input modalities: MLNN [32] operates on gate-level netlists, LSTM [33] and HTM [34] use power traces, and the LLM-based method [35] operates at the RTL/netlist level. Our method achieves competitive or superior active Trojan detection accuracy and is the only approach offering three-state classification with dormant detection capability.

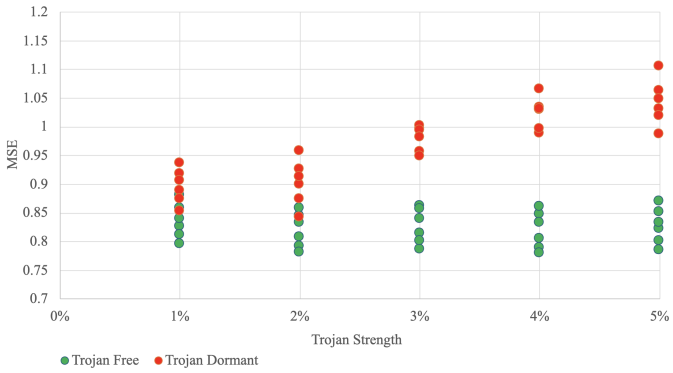


Fig. 3: Detection Sensitivity Thresholds of HTs

TABLE III: Comparison of HOODOO with Existing Methods

Method	**3-State Class?**	**Dormant Acc. (%)**	**Active Acc. (%)**
MLNN [32]	No	N/A	85.0
LSTM [33]	No	N/A	86.8
HTM [34]	No	N/A	92.2
LLM (GPT-4) [35]	No	N/A	Context-Free: 81.0 Contextual: 91.7
NCDE (Ours)	Yes	1% Thr: 55.7 2% Thr: 62.5 3% Thr: 80.2 4% Thr: 88.3 5% Thr: 92.2	1% Thr: 92.4 2% Thr: 94.6 3% Thr: 95.4 4% Thr: 99.3 5% Thr: 100.0

F. Limitations and Challenges

Our detection assumes dormant Trojans produce a measurable side-channel footprint; Trojans with power variations below 3% of peak amplitude may evade detection. Also, the sensitivity analysis uses Gaussian noise as a proxy; real dormant Trojans may exhibit structured, non-Gaussian signatures. Finally, MSE is the sole anomaly feature; richer residual features from NCDE latent states could improve sensitivity.

VI. Conclusion

In this paper, we presented a novel approach for HT detection and prediction, employing NCDEs. The proposed approach enables effective three-state classification, distinguishing between no Trojan, dormant Trojan, and active Trojan conditions to support proactive security measures. Experimental validation demonstrates superior performance relative to conventional machine learning techniques. The core framework, coupling NCDEs with temporal analysis, shows significant potential for broader applications within hardware security and reliability, including but not limited to, side-channel attack mitigation, device aging monitoring, and fault detection paradigms.

Future research directions include integrating multi-modal side-channel data to improve the sensitivity of dormant Trojan detection, exploring richer anomaly features from NCDE latent states beyond MSE, evaluating generalization across diverse chip designs and process variation conditions, and applying transfer learning techniques to reduce training data requirements for new hardware designs.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Award No. 2245247.

REFERENCES

[1] M. Xue, C. Gu, W. Liu, S. Yu, and M. O'Neill, "Ten years of hardware Trojans: A survey from the attacker's perspective," In *IET Computers & Digital Techniques*, vol. 14, pp. 231–246, 2020.

[2] J. Francq and F. Frick, "Introduction to hardware Trojan detection methods," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 770–775, 2015.

[3] H. Wang, Q. Zhou, and Y. Cai, "Static probability analysis guided RTL hardware Trojan test generation," In *Asia and South Pacific Design Automation Conference*, pp. 510–515, 2023.

[4] Z. Pan and P. Mishra, "Automated test generation for hardware Trojan detection using reinforcement learning," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 408–413, 2021.

[5] J. Clements and Y. Lao, "Hardware Trojan design on neural networks," In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-5, 2019.

[6] R. Lu, H. Shen, Y. Su, H. Li, and X. Li, "GramsDet: Hardware Trojan detection based on recurrent neural network," In *IEEE Asian Test Symposium (ATS)*, pp. 111–115, 2019.

[7] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," In *IEEE Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[8] S. Yang, P. Chakraborty, and S. Bhunia, "Side-channel analysis for hardware Trojan detection using machine learning," in *IEEE International Test Conference India (ITC India)*, pp. 1–6, 2021.

[9] A. Hepp, J. Baehr, and G. Sigl, "Golden model-free hardware Trojan detection by classification of netlist module graphs," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1317–1322, 2022.

[10] T. Perez and S. Pagliarini, "A side-channel hardware Trojan in 65nm CMOS with 2μW precision and multi-bit leakage capability," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 9–10, 2022.

[11] R. Vishwakarma and A. Rezaei, "Risk-aware and explainable framework for ensuring guaranteed coverage in evolving hardware Trojan detection," In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–9, 2023.

[12] Z. Wang, Y. Xue, and H. Wang, "Secure run-time hardware Trojan detection using lightweight analytical models," In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 1–9, 2023.

[13] S. Kaji, D. Fujimoto, and Y. Hayashi, "Simulation-based approach to generating golden data for PCB-level hardware Trojan detection using capacitive sensor," In *IEEE Physical Assurance and Inspection of Electronics (PAINE)*, pp. 1–7, 2023.

[14] R. Vishwakarma and A. Rezaei, "Uncertainty-aware hardware Trojan detection using multimodal deep learning," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2024.

[15] R. Vishwakarma and A. Rezaei, "Uncertainty-aware unimodal and multimodal learning for evolving hardware Trojan detection," In *Journal of Hardware and Systems Security*, Vol. 9, pp. 1–23, 2025.

[16] K. Abe, D. Fujimoto and Y. Hayashi, "Fundamental study on detecting hardware Trojans in printed circuit boards using ring oscillators," In *International Workshop on the Electromagnetic Compatibility of Integrated Circuits (EMC Compo)*, pp. 1–4, 2024.

[17] R. Yasaei, L. Chen, S. -Y. Yu, and M. Abdullah Al Faruque, "Hardware Trojan detection using graph neural networks," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 1, pp. 25–38, 2025.

[18] T. Ishikawa, K. Yokooji, Y. Midoh, N. Miura, M. Shintani, and J. Shiomi, "Hardware Trojan detection by fine-grained power domain partitioning," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 1257–1263, 2025.

[19] J. Maynard and A. Rezaei, "Reconfigurable run-time hardware Trojan mitigation for logic-locked circuits," In *IEEE 17th Dallas Circuits and Systems Conference (DCAS)*, pp. 1–6, 2024.

[20] L. Chen, Y. Gamal, Y. Li, S. -Y. Yu, I. Alouani, and M. A. A. Faruque, "DART: Distribution-aware hardware Trojan detection," In *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 9600–9609, 2025.

[21] R. Kumar Kundu, K. Khalil, E. Garcia, E. Grassia, P. Calyam, and K. A. Hoque, "PEARL: An adaptive and explainable hardware Trojan detection using open source and enterprise large language models," In *IEEE Access*, vol. 13, pp. 133755–133772, 2025.

[22] B. Li, C. Dong, D. Qiu, M. Chen, and Y. Yang, "Defense in the reverse fragment: RL-based partial netlist hardware Trojan detection," In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–7, 2025.

[23] Z. Pan, Z. Shu and X. Yu, "SAGE: Shapley attention graph network for gate-level Trojan detection and localization," In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 1–6, 2025.

[24] W. Hu, B. Li, L. Wu, Y. Li, X. Li, and L. Hong, "Design for assurance: Employing functional verification tools for thwarting hardware Trojan threat in 3PIPs," In *IEEE Transactions on Dependable and Secure Computing*, vol. 23, no. 1, pp. 132–148, 2026.

[25] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," In *International Conference on Neural Information Processing Systems (NIPS)*, pp. 6696–6707, 2020.

[26] P. Ma, Z. Wang, and Y. Wang, "A pre-silicon detection based on deep learning model for hardware Trojans," In *Journal of Circuits Systems and Computers*, vol. 33, no. 8, 2024.

[27] E. Puschner, T. Moos, S. Becker, C. Kison, A. Moradi, and C. Paar, "Red team vs. blue team: A real-world hardware Trojan detection case study across four modern CMOS technology generations," In *IEEE Symposium on Security and Privacy (SP)*, pp. 56–74, 2023.

[28] T. Mosavirik, S. K. Monfared, M. S. Safa, and S. Tajik, "Silicon echoes: Non-invasive Trojan and tamper detection using frequency-selective impedance analysis," In *Cryptology ePrint Archive*, Paper 2023/075, 2023.

[29] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," In *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[30] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pp. 51–57, 2008.

[31] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," In *IEEE Symposium on Security and Privacy (SP)*, pp. 296–310, 2007.

[32] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," In *IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 227–232, 2017.

[33] A. Nasr, K. Mohamed, A. Elshenawy, and M. Zaki, "A Siamese deep learning framework for efficient hardware Trojan detection using power side-channel data," In *Scientific Reports*, vol. 14, no. 13013, pp. 1–13, 2024.

[34] S. Faezi, R. Yasaei, A. Barua, and M. A. A. Faruque, "Brain-inspired golden chip free hardware Trojan detection," In *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2697–2708, 2021.

[35] D. Saha, S. Tarek, K. Yahyaei, S. K. Saha, J. Zhou, M. Tehranipoor, and F. Farahmandi, "LLM for SoC security: A paradigm shift," In *IEEE Access*, vol. 12, pp. 155498–155521, 2024.

[36] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

[37] J. Morrill, P. Kidger, C. Salvi, J. Foster, and T. Lyons, "Neural rough differential equations for long time series," In *International Conference on Machine Learning (PMLR)*, pp. 7829–7838, 2021.

[38] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, "GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series," In *Advances in Neural Information Processing Systems (NIPS)*, pp. 7379–7390, 2019.

[39] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons learned after one decade of research," In *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 1, pp. 1–23, 2016.

[40] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," In *Annual International Cryptology Conference*, pp. 388–397, 1999.

[41] J. Morrill, P. Kidger, L. Yang, and T. Lyons, "Neural controlled differential equations for online prediction tasks," In *arXiv preprint arXiv:2106.11028*, 2021.

[42] R. Yasaei, S. Faezi, and M. A. A. Faruque, "Hardware Trojan power & EM side-channel dataset," In *IEEE Dataport*, https://dx.doi.org/10.21227/9fwb-8978.

[43] B. Omidi, K. N. Khasawneh, and I. Alouani, "Evasive hardware Trojan through adversarial power trace," arXiv preprint arXiv:2401.02342, 2024.