CirSTAG: Circuit Stability Analysis on Graph-based Manifolds

Wuxinlin Cheng
Stevens Institute of Technology
wcheng7@stevens.edu

Yihang Yuan
Stevens Institute of Technology
yyuan22@stevens.edu

Chenhui Deng
NVIDIA Research
cdeng@nvidia.com

Ali Aghdaei
University of California, San Diego
aaghdaei@ucsd.edu

Zhiru Zhang
Cornell University
zhiruz@cornell.edu

Zhuo Feng
Stevens Institute of Technology
zfeng12@stevens.edu

***Abstract*—Circuit stability (sensitivity) analysis aims to estimate the overall performance impact of variations in underlying design parameters, such as gate sizes and capacitance. This process is challenging because it often requires numerous time-consuming circuit simulations. In contrast, graph neural networks (GNNs) have shown remarkable effectiveness and efficiency in tackling several chip design automation issues, including circuit timing predictions, parasitic prediction, gate sizing, and device placement. This paper introduces a novel approach called CirSTAG, which utilizes GNNs to analyze the stability (robustness) of modern integrated circuits (ICs). CirSTAG is grounded in a spectral framework that examines the stability of GNNs by leveraging input/output graph-based manifolds. When two adjacent nodes on the input manifold are mapped (through a GNN model) to two remote nodes (data samples) on the output manifold, this indicates a significant mapping distortion (DMD) and consequently poor GNN stability. CirSTAG calculates a stability score equivalent to the local Lipschitz constant for each node and edge, considering both graph structure and node feature perturbations. This enables the identification of the most critical (sensitive) circuit elements that could significantly impact circuit performance. Our empirical evaluations across various timing prediction tasks with realistic circuit designs demonstrate that CirSTAG can accurately estimate the stability of each circuit element under diverse parameter variations. This offers a scalable method for assessing the stability of large integrated circuit designs.**

I. Introduction

Stability analysis is crucial in the design and optimization of Very Large Scale Integration (VLSI) circuits, ensuring reliable performance under perturbations such as design parameter changes (e.g., transistor sizing), process and temperature variations, and supply voltage fluctuations [1]. Assessing circuit stability is essential to prevent unintended behaviors like oscillations or unpredictable outputs, which can adversely impact the functionality and reliability of integrated circuits (ICs). Moreover, stability analysis guides circuit optimization tasks, such as gate sizing for timing and power optimization [21], by identifying the most unstable circuit nodes that, when modified, can significantly improve overall performance. However, existing stability analysis methods often require numerous repeated circuit simulations after perturbing underlying parameters, becoming prohibitively expensive for large-scale designs.

Graph Neural Networks (GNNs) have emerged as powerful tools in machine learning, particularly for graph-structured data [26], [32]. By integrating graph structures and node features, GNNs produce low-dimensional embedding vectors that preserve graph structural information [16]. They have been successfully applied in various real-world applications, including recommendation systems [14], traffic flow prediction [29], device placement [23], and VLSI timing prediction [17].

In this work, we present **CirSTAG**, a novel framework to quantify the stability of circuit networks leveraging GNNs. For a pre-trained GNN model that mimics the behaviors of a family of circuit designs (e.g., pre-routing slack prediction [17]), CirSTAG assesses circuit stability by estimating the impact of potential input perturbations (e.g., graph topology and node feature modifications) on the GNN's output. Specifically, if two nearby circuit nodes on the input graph are mapped to distant embedding vectors by the GNN, it indicates a substantial *distance mapping distortion* (DMD) and poor circuit stability. To address this, we propose an efficient spectral framework for quantifying DMDs of GNNs on input/output graph-based manifolds. We also introduce a stability score for each node (or edge) computed based on the DMD estimation, which is equivalent to the local Lipschitz constant of the GNN model.

CirSTAG enjoys near-linear runtime complexity and is compatible with various GNN architectures due to its data-centric nature. Experimental results demonstrate CirSTAG's capability to gauge individual node stability within diverse GNN models for realistic circuit designs.

Our key contributions are summarized as follows:

- To the best of our knowledge, we introduce the first spectral framework for analyzing circuit stability by evaluating node-level stability in GNNs, achieved by measuring the DMD of adjacent nodes using input/output graph-based manifolds.
- We construct proper graph-based manifolds for estimating DMDs by exploiting probabilistic graphical models (PGMs) and spectral graph embedding, preserving key structural properties of the given graph dataset.

- CirSTAG has near-linear time complexity and is agnostic to GNN models and node label information. Our empirical results validate that our approach provides a scalable method for assessing individual node (gate) stability in pre-routing timing prediction [17] and reverse engineering of realistic circuit designs [4].

The rest of the paper is organized as follows. In Section II, we provide background on concepts related to DMD and PGM. Section III offers an overview of the proposed CirSTAG framework. In Section IV, we introduce the technical details of CirSTAG, including its algorithm flow and complexity. In Section V, we present experimental results evaluating the performance of CirSTAG on stability analysis tasks using real-world VLSI design benchmarks. Finally, we conclude the paper in Section VI.

II. Background

A. Applications of GNNs in Electronic Design Automation

GNNs have demonstrated significant potential in Electronic Design Automation (EDA), revolutionizing tasks ranging from circuit design to verification by modeling complex relationships within electronic circuits.

In circuit analysis, GNNs enable efficient modeling and simulation of large-scale circuits, leading to accurate predictions of circuit behavior under various conditions [3]. For optimization, they facilitate the identification of optimal circuit configurations [24]. GNNs also aid in fault detection and diagnosis by pinpointing potential faults within electronic systems and suggesting corrective measures, thus improving reliability and reducing maintenance costs [7].

In layout and routing, GNNs manage the complexity of modern integrated circuits by optimizing layouts for better performance and lower power consumption while ensuring fabrication constraints are met [23]. Additionally, they have been applied to timing analysis, predicting arrival times and slack at timing endpoints, enhancing the efficiency and accuracy of EDA processes [17].

Furthermore, GNNs have emerged as transformative tools in circuit reverse engineering and security. By deciphering relationships between circuit components, GNNs facilitate the extraction and reconstruction of circuit functionalities from physical layouts. This capability is crucial for understanding proprietary designs, detecting unauthorized alterations, and ensuring compliance with intellectual property laws. Recent advancements demonstrate that GNNs can accurately identify sub-circuits and their functionalities, advancing the field of hardware security and trust [4], [28].

B. Manifold-Based Stability Analysis of ML Models

The stability of the ML model refers to its ability to produce consistent outputs despite small variations or noise in the input [25]. Let F denote an ML model that operates on input X to yield output Y, i.e., $Y = F(X)$. The stability of F can be evaluated by measuring distortions in the mapping between low-dimensional input and output manifolds. Specifically, the *distance mapping distortion* (DMD) metric quantifies distance

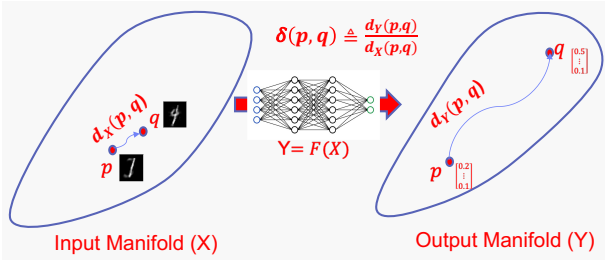

Fig. 1: Distance mapping distortions on manifolds [8]

distortions between graph-based manifolds [8], as illustrated in Fig. 1. For two input data samples p and q, the DMD metric $\delta^F(p,q)$ is defined as the ratio of the distance $d_Y(p,q)$ on the output manifold $G_Y = (V, E_Y)$ to the distance $d_X(p,q)$ on the input manifold $G_X = (V, E_X)$:

$$\delta^F(p,q) \stackrel{\text{def}}{=} \frac{d_Y(p,q)}{d_X(p,q)}. \tag{1}$$

Evaluating DMD between pairs of data samples helps determine the stability of the ML model. If two data samples that are close on the input manifold are mapped to distant points on the output manifold (i.e., $\delta^F(p,q)$ is large), this indicates a large local Lipschitz constant and poor stability of the model near those samples.

C. Graph Topology Learning via PGM

Given M samples of N-dimensional vectors stored in a data matrix $X \in \mathbb{R}^{N\times M}$, recent graph topology learning methods [11]–[13] estimate the graph structure (specifically, the Laplacian matrix L) from X to achieve two main objectives: **(1) Signal Smoothness over the Graph:** This can be quantified by the matrix trace $\mathrm{Tr}(X^\top L X)$ [18]. **(2) Sparsity of the Estimated Graph Topology:** Encouraging sparsity leads to simpler and more interpretable graph structures. Consider a random vector $x \sim \mathcal{N}(0, \Sigma)$ with probability density function:

$$f(x) = \frac{\exp\left(-\frac{1}{2}x^\top \Sigma^{-1} x\right)}{(2\pi)^{N/2}\det(\Sigma)^{1/2}} \propto \det(\Theta)^{1/2}\exp\left(-\frac{1}{2}x^\top \Theta x\right), \tag{2}$$

where $\Sigma = \mathbb{E}[xx^\top] \succ 0$ is the covariance matrix, and $\Theta = \Sigma^{-1}$ is the precision matrix (inverse covariance matrix). When the sample covariance matrix S is obtained from M i.i.d. samples $X = [x_1, \ldots, x_M]$ drawn from $\mathcal{N}(0, \Sigma)$, each element $\Theta_{i,j}$ of the precision matrix encodes the conditional dependence between variables X_i and X_j. Specifically, $\Theta_{i,j} = 0$ implies that variables X_i and X_j are conditionally independent, given the rest.

The maximum likelihood estimation (MLE) of the precision matrix Θ (i.e., learning the Probabilistic Graphical Model) can be formulated as the following convex optimization problem [12], [19]:

$$\max_{\Theta} \quad F(\Theta) = \log\det(\Theta) - \frac{1}{M}\mathrm{Tr}(X^\top \Theta X), \tag{3}$$

subject to $\Theta = L + \frac{1}{\sigma^2}I$, where L is a valid graph Laplacian matrix, I is the identity matrix, and $\sigma^2 > 0$ is a prior feature variance.

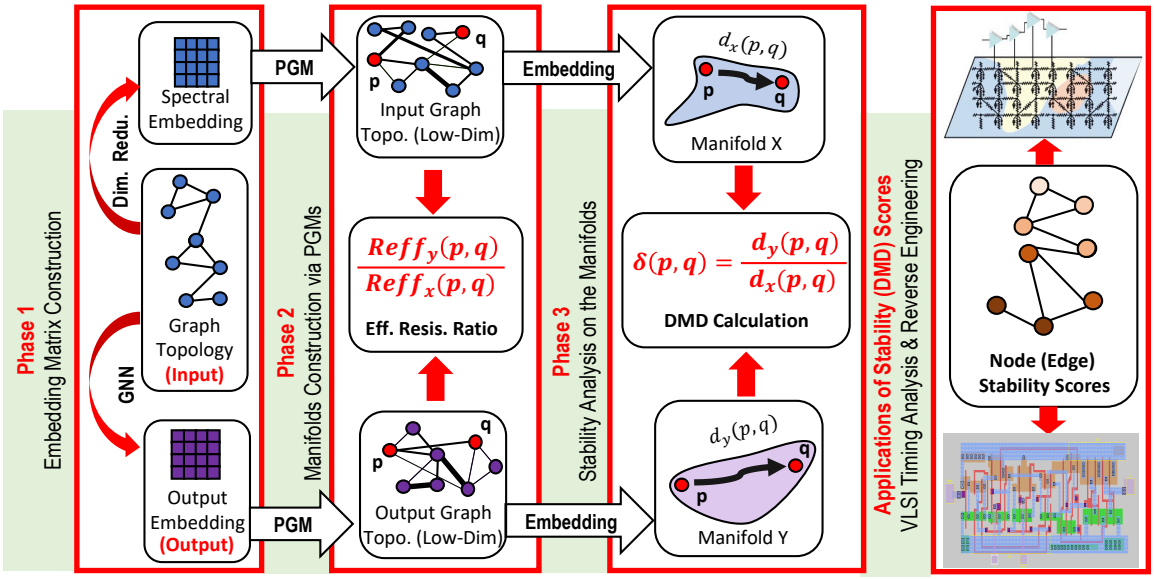

Fig. 2: The proposed GNN-based circuit stability analysis framework (CirSTAG) and its applications.

III. Overview of CirSTAG

To quantify the stability of an ML model, the DMD metric has been introduced [8] to provide an effective measure of how an ML model distorts pairwise distances between data samples. However, applying DMD to GNNs presents several challenges. DMD assumes that both input and output data can be well-represented by low-dimensional graph-based manifolds [8]. While this assumption typically holds for GNN outputs (i.e., node embedding vectors), it often does not for input circuit graphs, which may reside in relatively high-dimensional spaces. As shown in Fig. 4, directly using the original circuit networks (graphs) as the input manifold fails to accurately predict circuit delay stability under node feature (e.g., capacitance) perturbations.

Fig. 2 presents an overview of the proposed **CirSTAG** framework and its applications in design automation tasks. Our methodology consists of three distinct phases:

1) **Phase 1**: We construct an input embedding matrix by employing spectral graph embedding to retain the original graph's structural characteristics. We also obtain an output embedding matrix generated by the GNN model.
2) **Phase 2**: We create low-dimensional input and output graph-based manifolds using a highly scalable PGM construction algorithm.
3) **Phase 3**: We quantify node stability by computing DMDs that encapsulate the mappings (via GNNs) between the input and output graph-based manifolds.

In the following sections, we delve into the specifics of our CirSTAG framework.

IV. Methodology

A. Phase 1: Input/Output Embedding Matrices Construction

a) Nonlinear Dimensionality Reduction of the Input Graph: To reduce the dimensionality of the input graph for manifold construction, we employ a nonlinear dimensionality reduction approach inspired by the *Laplacian Eigenmap* algorithm [5]. The key idea is to construct a graph-based manifold from the high-dimensional data and then map each data sample (node) into a low-dimensional representation using the first few Laplacian eigenvectors. Specifically, given a graph with adjacency matrix A and degree matrix D, we compute the normalized graph Laplacian matrix $L_{\text{norm}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where I is the identity matrix. We then obtain the top M smallest eigenvalues $\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_M$ and their corresponding eigenvectors $\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_M$ of L_{norm}. We construct the input spectral embedding matrix to encapsulate key structural characteristics of the graph [10]:

$$U_M = \left[\sqrt{|1 - \tilde{\lambda}_1|}\, \tilde{u}_1, \ldots, \sqrt{|1 - \tilde{\lambda}_M|}\, \tilde{u}_M \right]. \quad (4)$$

This embedding matrix U_M effectively reduces the dimensionality of the input graph while preserving its essential structural properties. In the next phase, we will use U_M to construct the input graph-based manifold via graph topology learning.

b) Output Embedding from GNNs: Since the outputs of GNNs are typically low-dimensional node embeddings, we directly use the node embeddings generated by the GNN as the output embedding matrix. This output embedding matrix will similarly be transformed into an output graph-based manifold through graph topology learning in the subsequent phase.

B. Phase 2: Constructing Graph-Based Manifolds via PGMs

In this phase, we construct the input and output graph-based manifolds using the embedding matrices obtained in Phase 1. This process is motivated by a recent PGM-based graph topology learning framework (SGL) [15], [30]. However, the original SGL framework requires numerous iterations to converge, leading to superlinear runtime complexity. To enhance scalability, we introduce an efficient spectral sparsification scheme for learning PGMs. Consider a weighted undirected

graph $G = (V, E, w)$, with edge weights $w \in \mathbb{R}^V_{\geq 0}$ and $|V| = N$. The Laplacian matrix L of G can be expressed as:

$$L = \sum_{(p,q)\in E} w_{p,q} e_{p,q} e_{p,q}^\top, \tag{5}$$

where $e_{p,q} = e_p - e_q$, $w_{p,q}$ is the weight of edge (p, q), and $e_p \in \mathbb{R}^N$ is the standard basis vector with a 1 at the p-th entry. We aim to solve the following convex optimization problem [12]:

$$\begin{aligned} &\max_{\Theta} : F(\Theta) = F_1 - \frac{1}{M} F_2 \\ &F_1 = \log\det(\Theta) = \sum_{i=1}^{N} \log(\lambda_i + \frac{1}{\sigma^2}) \\ &F_2 = Tr(X^\top \Theta X) = \frac{Tr(X^\top X)}{\sigma^2} + \sum_{(p,q)\in E} w_{p,q} \|X^\top e_{p,q}\|_2^2 \end{aligned} \tag{6}$$

where $\Theta = L + \frac{1}{\sigma^2} I$, λ_i are the Laplacian eigenvalues, and $X \in \mathbb{R}^{N\times M}$ is the input data matrix equal to the spectral embedding matrix U_M from (4). To maximize $F(\Theta)$, we compute the partial derivatives with respect to $w_{p,q}$ [30]:

$$\begin{aligned} \frac{\partial F_1}{\partial w_{p,q}} &= R^{\text{eff}}_{p,q}, \\ \frac{\partial F_2}{\partial w_{p,q}} &= D^{\text{data}}_{p,q} = \|X^\top e_{p,q}\|_2^2 = \frac{1}{w_{p,q}}, \end{aligned} \tag{7}$$

where $R^{\text{eff}}_{p,q}$ is the effective resistance between nodes p and q, and $D^{\text{data}}_{p,q}$ is the ℓ_2 distance between their data vectors. Based on these gradients, we propose to prune non-critical edges from an initial dense graph by considering the spectral distortion metric [15]:

$$\eta_{p,q} = \frac{R^{\text{eff}}_{p,q}}{D^{\text{data}}_{p,q}} = w_{p,q} R^{\text{eff}}_{p,q}. \tag{8}$$

Edges with low effective resistance but large data distance (i.e., small $\eta_{p,q}$) are pruned. This strategy effectively maximizes F_2 without significantly decreasing F_1, allowing us to construct PGMs via spectral graph sparsification.

a) Initial Dense Graph Construction: Given the low-dimensional nature of the data from Phase 1, we efficiently construct the initial dense graph using the k-nearest neighbor (kNN) algorithm [22], which has computational complexity $O(|V| \log |V|)$.

b) PGM Refinement via Spectral Sparsification: To refine the PGM, we apply spectral graph sparsification using a short-cycle decomposition scheme [9]. This method partitions the graph into disjoint cycles by removing a fixed number of edges, ensuring cycles are bounded in length. Recent algorithms (Lemma 1) combine short-cycle decomposition with low-stretch spanning trees (LSSTs) to preserve the spectral properties of the original graph [20].

Lemma 1: Spectral sparsification of an unweighted, undirected graph G with Laplacian L_G can be achieved using a short-cycle decomposition algorithm, yielding a sparsified graph H with Laplacian L_H such that for all real vectors x, $x^\top L_G x \approx x^\top L_H x$ [9].

Extending prior work, we introduce a novel low-resistance-diameter (LRD) decomposition scheme to handle weighted graphs by restricting cycle lengths measured via effective resistance. Our method efficiently computes the effective resistance of each edge and uses a multilevel framework [2] to decompose the graph into cycles bounded by a given effective resistance threshold.

C. Phase 3: Stability Analysis on the Manifolds

To quantify the stability of the ML model, we utilize the DMD (as defined by equation (1)) [8], which measures how the model distorts pairwise distances between data samples. While geodesic distances are natural choices for d_X and d_Y, computing all-pairs geodesic distances is computationally prohibitive for large graphs [27]. To address this, we adopt the *effective resistance distance* [6], which can be efficiently computed and captures global graph structure. Suppose the input X is mapped to output Y by $Y = F(X)$. Let L_X and L_Y denote the Laplacian matrices of the input and output manifolds G_X and G_Y, respectively. Recent theoretical results [8] allow us to assess node stability using the largest generalized eigenvalues and corresponding eigenvectors of $L_Y^+ L_X$. We compute the weighted eigensubspace matrix $V_s \in \mathbb{R}^{N\times s}$: $V_s = \left[v_1\sqrt{\zeta_1},\ \ldots,\ v_s\sqrt{\zeta_s}\right]$, where $\zeta_1 \geq \zeta_2 \geq \cdots \geq \zeta_s$ are the largest s eigenvalues of $L_Y^+ L_X$, and $v_1, v_2, \ldots, v_s$ are the corresponding eigenvectors. Using V_s, we embed the input graph G_X, associating each node with an s-dimensional vector. The stability of an edge $(p, q) \in E_X$ is quantified by the embedding distance between its end nodes: $\|V_s^\top e_{p,q}\|_2^2$, where $e_{p,q} = e_p - e_q$, and e_p is the standard basis vector. The stability score of a node p is estimated as:

$$\begin{aligned} \textbf{Score}^F(p) &\stackrel{\text{def}}{=} \frac{1}{|\mathbb{N}_X(p)|} \sum_{q_i \in \mathbb{N}_X(p)} \left(\|V_s^\top e_{p,q_i}\|_2^2\right) \\ &\propto \frac{1}{|\mathbb{N}_X(p)|} \sum_{q_i \in \mathbb{N}_X(p)} \left(\delta^F(p, q_i)\right)^3, \end{aligned} \tag{9}$$

where $\mathbb{N}_X(p)$ denotes the set of neighbors of node p in G_X. This node stability score serves as a surrogate for the local Lipschitz constant $\|\nabla F(p)\|$ under the manifold setting [8].

D. Runtime Complexity

For spectral graph embedding, we can exploit fast multilevel eigensolvers that allow computing the first c Laplacian eigenvectors in nearly-linear time $O(c|V|)$ without loss of accuracy [31]. Then, the k-nearest neighbor algorithm [22] computational complexity is $O(|V| \log |V|)$. Sparsification via short-cycle decomposition has $O(|V|dm)$ time complexity. By leveraging fast Laplacian solvers [31], all nodes' stability scores can be computed with $O(|E|)$, where the V/E denotes the number of nodes/edges, d is the average degree of the matrix, and m is the order of Krylov subspace. The proposed CirSTAG algorithm flow is shown in Algorithm 1.

Algorithm 1 The algorithm flow of CirSTAG

Input: Input graph G, output Y
Output: Node stability scores in (9).

1: $U_N \leftarrow$ compute_weighted_spectral_embedding(G)
2: $kNN\_dense\_graph \leftarrow$ construct_kNN_graph(U_N)
3: $input\_PGM \leftarrow$ sparse_graph(kNN_dense_graph)
4: $kNN\_dense\_graph\_Y \leftarrow$ construct_kNN_graph(Y)
5: $output\_PGM \leftarrow$ sparse_graph$(kNN_dense_graph_Y)$
6: $L_X \leftarrow$ compute_laplacian$(input_PGM)$
7: $L_Y \leftarrow$ compute_laplacian$(output_PGM)$
8: $V_N \leftarrow$ calculate_generalized_eigenvectors(L_Y, L_X)
9: **for** each node p in G_X **do**
10: Compute its stability score by:

$$\textbf{Score}(p) = \frac{1}{|\mathbb{N}_X(p)|} \sum_{q_i \in \mathbb{N}_X(p)} \left(\|V_s^\top e_{p,q_i}\|_2^2 \right)$$

11: **end for**

V. Experimental Results

To demonstrate the capability of **CirSTAG** for stability analysis in EDA applications, we present two case studies using high-accuracy GNN models: **(A)** Circuit Timing Prediction [17] and **(B)** Circuit Functional Reverse Engineering [4]. The effectiveness of CirSTAG relies on accurate predictions from the GNN models, which serve as black-box simulators for their respective tasks. Accurate embeddings ensure that the computed DMDs reflect true circuit behavior, allowing CirSTAG to capture performance variations due to node feature or topology perturbations. In Case Study (A), the GNN predicts signal arrival times (delays) at each circuit node, focusing on **node feature perturbations**. In Case Study (B), we examine the impact of **topology perturbations** on the cosine similarity of embeddings and the F1 macro score for multi-task classification of sub-circuits. These studies validate our methodology by assessing node stability under different perturbations.

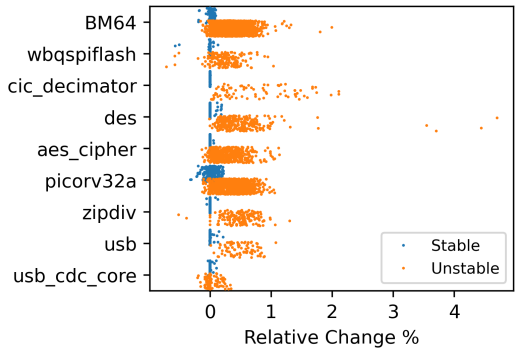

Fig. 3: Circuit delay variations (with dimension reduction).

A. Stability Analysis of Circuit Delays under Node Feature Perturbations

In this case study, we analyze circuit delay stability considering node feature (pin capacitance) perturbations using datasets and a GNN model from [17]. The GNN predicts pre-routing static timing analysis (STA) results, interpreting graphs where nodes represent cell pins and edges represent net connections and internal cell connections. We selected nine circuit datasets with the highest R^2 scores (96.88% to 99.22%) to ensure accurate GNN predictions. The GNN then predicted arrival times at primary output pins, and we computed the relative output change compared to unperturbed data. Nodes representing output pins were excluded, as they do not directly affect internal timing dynamics. Table I summarizes the relative changes in arrival time predictions. The results consistently show that perturbing unstable nodes leads to higher average and maximum relative changes compared to stable nodes, indicating greater sensitivity to capacitance changes. Increasing the scaling factor from 5 to 10 nearly doubles the relative change. However, increasing the percentage of perturbed nodes from 5% to 15% does not proportionally increase the relative change for unstable nodes, suggesting that the most unstable nodes contribute more significantly to variability. Fig. 3 illustrates the distribution of relative changes when perturbing the top 10% of nodes with a scaling factor of 10, highlighting the contrast between unstable and stable nodes.

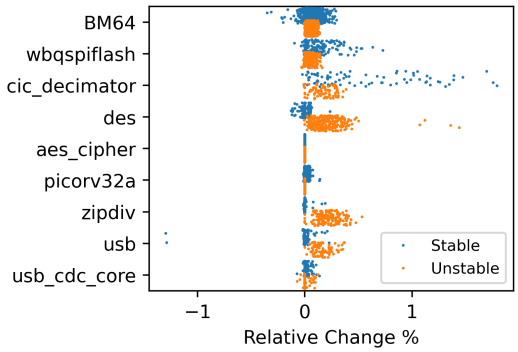

Fig. 4: Circuit delay variations(without dimension reduction).

Ablation Study of CirSTAG We conducted an ablation study by omitting the graph dimensionality reduction stage in CirSTAG. As shown in Fig. 4, the distribution of relative changes becomes more random without dimension reduction, implying that this stage is crucial for accurate instability ranking.

Scalability Analysis of CirSTAG Fig. 5 shows the runtime of CirSTAG across the nine benchmarks of varying complexity. CirSTAG demonstrates near-linear runtime, efficiently handling large circuit designs.

B. Stability Analysis under Circuit Topology Perturbations

In this case study, we investigate topology perturbations to demonstrate that **CirSTAG** effectively captures topology-related stability in functional reverse engineering tasks. We utilize datasets and a GNN model from [4], which employs a Graph Attention Network (GAT) for sub-circuit identification and labeling, achieving an accuracy of 98.87% on the interconnected dataset. In this model, graphs are derived from netlists, with nodes representing gates and edges depicting gate connections. Node features include surrounding gate information, detailing Boolean functionalities from gate inputs in the local neighborhood. Direct topology perturbations could mismatch

TABLE I: Comparison of average and maximum relative change (%) in arrival time prediction at primary output pin. Results for unstable and stable nodes are presented separately, denoted as 'unstable/stable'.

benchemark	Circuit stability analysis by CirSTAG with a GNN-based pre-routing timing analysis tool [17]											
	scale factor = 5×						scale factor = 10×					
	node perturb 5%		node perturb 10%		node perturb 15%		node perturb 5%		node perturb 10%		node perturb 15%	
	mean	max	mean	max	mean	max	mean	max	mean	max	mean	max
BM64	0.0915/ 0.0006	0.6328/ 0.0414	0.1391/ 0.0005	0.9016/ 0.0414	0.1563/ 0.0005	0.9828/ 0.0414	0.2057/ 0.0013	1.3715/ 0.0934	0.3125/ 0.0012	1.9934/ 0.0934	0.3504/ 0.0012	2.1751/ 0.0934
wbqspiflash	0.1106/ 0.0000	0.4078/ 0.0012	0.1090/ -0.0021	0.4798/ 0.0578	0.1081/ -0.0009	0.4798/ 0.0699	0.2479/ 0.0000	0.8934/ 0.0027	0.2443/ -0.0048	1.0382/ 0.1299	0.2434/ -0.0021	1.0382/ 0.1574
cic_decimator	0.3273/ 0.0000	0.9271/ 0.0000	0.3304/ 0.0000	0.9271/ 0.0029	0.3305/ 0.0022	0.9266/ 0.1047	0.7355/ 0.0000	2.1081/ 0.0000	0.7505/ 0.0001	2.1081/ 0.0069	0.7506/ 0.0049	2.1080/ 0.2361
des	0.1886/ 0.0030	1.4600/ 0.0848	0.2300/ 0.0030	1.9617/ 0.0848	0.2318/ 0.0030	1.9617/ 0.0848	0.4620/ 0.0068	3.4748/ 0.1907	0.5211/ 0.0068	4.6989/ 0.1907	0.5250/ 0.0068	4.6989/ 0.1907
aes_cipher	0.0753/ 0.0002	0.4464/ 0.0392	0.1265/ 0.0002	0.5755/ 0.0392	0.1665/ 0.0002	0.6023/ 0.0392	0.1698/ 0.0005	1.0109/ 0.0884	0.2839/ 0.0005	1.1368/ 0.0884	0.3744/ 0.0005	1.1921/ 0.0884
picorv32a	0.1505/ 0.0023	0.4522/ 0.1000	0.1540/ 0.0028	0.4781/ 0.1000	0.1559/ 0.0033	0.4781/ 0.1489	0.3383/ 0.0051	1.0095/ 0.2197	0.3461/ 0.0062	1.0587/ 0.2266	0.3501/ 0.0074	1.0587/ 0.3304
zipdiv	0.2136/ 0.0000	0.5189/ 0.0000	0.2202/ 0.0003	0.5807/ 0.0572	0.2211/ 0.0003	0.5807/ 0.0572	0.4769/ 0.0000	1.1610/ 0.0000	0.4905/ 0.0007	1.3093/ 0.1279	0.4924/ 0.0006	1.3039/ 0.1279
usb	0.2069/ 0.0029	0.4066/ 0.0737	0.2182/ 0.0043	0.4782/ 0.0945	0.2254/ 0.0096	0.4782/ 0.2194	0.4696/ 0.0065	0.9178/ 0.1673	0.4949/ 0.0099	1.0773/ 0.2142	0.5107/ 0.0217	1.0773/ 0.4862
usb_cdc_core	0.0090/ 0.0006	0.0894/ 0.0321	0.0165/ 0.0014	0.1550/ 0.0471	0.0198/ 0.0013	0.2164/ 0.0460	0.0196/ 0.0013	0.2031/ 0.0724	0.0373/ 0.0031	0.3448/ 0.1057	0.0449/ 0.0029	0.4879/ 0.1032

TABLE II: Comparison of average F1 macro score for sub-circuit classification under different level of perturbation ratio.

benchmarks	# nodes	# edges	F1 macro score						
			0%	5%		10%		20%	
			Original	Unstable	Stable	Unstable	Stable	Unstable	Stable
add mul combine	806	1640	0.9961	0.9555	0.9961	0.9184	0.9961	0.7002	0.9961
add mul comp	926	1829	1.000	0.9702	0.9961	0.9050	0.9963	0.5960	0.9961
add_mul_comp_sub	1161	2248	0.9864	0.9521	0.9809	0.9261	0.9774	0.8557	0.9673
add_mul_mix	1637	3299	0.6135	0.6044	0.6135	0.5789	0.6135	0.5323	0.6135
add_mul_sub	1111	2170	0.9748	0.9486	0.9712	0.9421	0.9691	0.7663	0.9656
add_mul	863	1709	0.9320	0.9001	0.9320	0.8690	0.9320	0.7870	0.9320

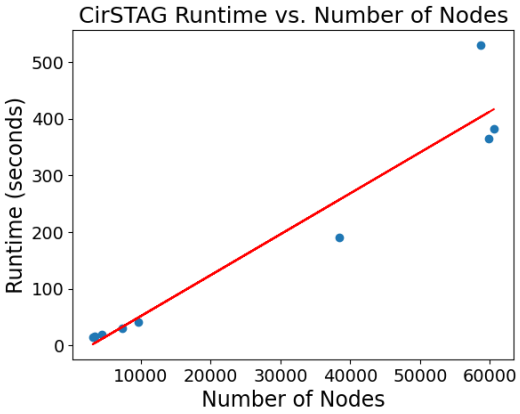

Fig. 5: Runtime scalability of CirSTAG

node features incorporating topological data. To address this, we perturb the topology while preserving feature relevance by clustering selected unstable or stable nodes based on input degrees and randomly swapping their output connections. This ensures that input topology and features remain valid post-perturbation. Analyzing the impact on sub-circuit classification using the F1 macro score, shown in Table II, we observe that as perturbation ratios increase, F1 macro scores decline more sharply for unstable nodes than for stable nodes. This suggests that unstable nodes, relying more on neighboring information, are more sensitive to topology changes, while stable nodes possess unique features supporting robust classification.

VI. Conclusion

We presented **CirSTAG**, a novel framework for quantifying the stability of circuit networks using GNNs. CirSTAG assesses circuit stability by estimating the impact of potential input perturbations on the GNN's output. By leveraging both input and output graph-based manifolds, our efficient spectral framework quantifies DMDs and assigns stability scores to nodes and edges, equivalent to the local Lipschitz constants of the GNN model. Empirical evaluations on pre-routing timing prediction and functional reverse engineering tasks demonstrate that CirSTAG effectively and scalably estimates circuit stability under various node feature and graph topology perturbations.

VII. Acknowledgments

This work is supported in part by the National Science Foundation under Grants CCF-2417619, CCF-2212370, and CCF-2205572.

REFERENCES

[1] Modeling and analysis of manufacturing variations. In *Proceedings of the IEEE 2001 Custom Integrated Circuits Conference (Cat. No. 01CH37169)*, pages 223–228. IEEE, 2001.

[2] A. Aghdaei and Z. Feng. Hyperef: Spectral hypergraph coarsening by effective-resistance clustering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

[3] L. Alrahis, J. Knechtel, and O. Sinanoglu. Graph neural networks: A powerful and versatile tool for advancing design, reliability, and security of ics. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pages 83–90, 2023.

[4] L. Alrahis, A. Sengupta, J. Knechtel, S. Patnaik, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu. Gnn-re: Graph neural networks for reverse engineering of gate-level netlists. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(8):2435–2448, 2022.

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[6] F. Chen, S. Roch, K. Rohe, and S. Yu. Estimating graph dimension with cross-validated eigenvalues. *arXiv preprint arXiv:2108.03336*, 2021.

[7] Z. Chen, J. Xu, C. Alippi, S. X. Ding, Y. Shardt, T. Peng, and C. Yang. Graph neural network-based fault diagnosis: a review. *arXiv preprint arXiv:2111.08185*, 2021.

[8] W. Cheng, C. Deng, Z. Zhao, Y. Cai, Z. Zhang, and Z. Feng. Spade: A spectral method for black-box adversarial robustness evaluation. In *International Conference on Machine Learning*, pages 1814–1824. PMLR, 2021.

[9] T. Chu, Y. Gao, R. Peng, S. Sachdeva, S. Sawlani, and J. Wang. Graph sparsification, spectral sketches, and faster resistance computation via short cycle decompositions. *SIAM Journal on Computing*, (0):FOCS18–85, 2020.

[10] C. Deng, X. Li, Z. Feng, and Z. Zhang. Garnet: Reduced-rank topology learning for robust and scalable graph neural networks. *arXiv preprint arXiv:2201.12741*, 2022.

[11] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016.

[12] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[13] H. E. Egilmez, E. Pavez, and A. Ortega. Graph learning from data under laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

[14] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[15] Z. Feng. Sgl: Spectral graph learning from measurements. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 727–732. IEEE, 2021.

[16] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[17] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin. A timing engine inspired graph neural network model for pre-routing slack prediction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1207–1212, 2022.

[18] V. Kalofolias. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pages 920–929, 2016.

[19] B. Lake and J. Tenenbaum. Discovering structure by learning sparse graphs. 2010.

[20] Y. P. Liu, S. Sachdeva, and Z. Yu. Short cycles via low-diameter decompositions. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2602–2615. SIAM, 2019.

[21] Y.-C. Lu, S. Nath, V. Khandelwal, and S. K. Lim. Rl-sizer: Vlsi gate sizing for timing optimization using deep reinforcement learning. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 733–738. IEEE, 2021.

[22] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[23] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[24] H. Ren, S. Nath, Y. Zhang, H. Chen, and M. Liu. Why are graph neural networks effective for eda problems? In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–8, 2022.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[27] R. R. Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM Journal on Computing*, 47(5):1965–1985, 2018.

[28] N. Wu, Y. Li, C. Hao, S. Dai, C. Yu, and Y. Xie. Gamora: Graph learning based symbolic reasoning for large-scale boolean networks. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[29] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[30] Y. Zhang, Z. Zhao, and Z. Feng. Sf-sgl: Solver-free spectral graph learning from linear measurements. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[31] Z. Zhao, Y. Zhang, and Z. Feng. Towards scalable spectral embedding and data visualization via spectral coarsening. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 869–877, 2021.

[32] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.