Explainable GNN-Driven Test Point Insertion on Uncontrollable I/Os

Sung-Hyuk Cho[1], Tae-Min Park[1], Jeongyeol Lee[1,3], Jae-Youn Hong[1], Andreas Gerstlauer[4], and Joon-Sung Yang[1,2]
[1]Department of Electrical and Electronic Engineering, [2]Department of Systems Semiconductor Engineering,
Yonsei University, Seoul, South Korea
[3]System LSI Division, Samsung Electronics, Hwaseong, South Korea
[4]Department of Electrical and Computer Engineering, The University of Texas, Austin, TX, USA

***Abstract*—Test coverage degradation from uncontrollable I/Os is a critical challenge in modern SoC design. In area-sensitive applications, such as the peripheral circuits of memory devices, standard DFT solutions like wrapper chains are prohibitively expensive due to their high area overhead. This necessitates a surgical Test Point Insertion (TPI) strategy that maximizes testability while adhering to strict cost constraints. To address this challenge, we propose a novel TPI framework using an explainable Graph Neural Network (GNN). Our GNN accurately predicts test coverage in circuits with masked I/Os, and an integrated saliency map (XAI) technique then identifies the most critical I/Os for TPI. Compared to a leading commercial tool, our framework achieves the target test coverage with 7.53% fewer TPs and improves coverage by 4.34% with the same TP budget on average. The scalability on large circuits (>100k gates) and technology independence confirm its practical applicability for minimizing die cost in constrained, real-world designs.**

***Index Terms*—Test Point Insertion, Graph Neural Networks, Saliency Map, Uncontrollable IO, Explainable AI**

I. Introduction

Modern System-on-Chip (SoC) integration frequently involves hardened IPs, introducing uncontrollable inputs/outputs (I/Os). These I/Os pose a significant challenge to design-for-test (DFT), causing a loss in test coverage (TC) that standard ATPG tools struggle to recover [1]. Standard DFT solutions like full boundary scan or wrapper chains, while comprehensive, introduce significant area and performance overhead. In high-volume, cost-sensitive applications – especially the peripheral circuits of modern memory devices – such overhead often exceeds tight design budgets and inflates die cost, making full-scale DFT rarely viable [2].

This creates a critical design trade-off: recovering TC while adhering to extremely strict area constraints. The most practical solution is a highly selective Test Point Insertion (TPI) strategy, where the TPs are placed only on the subset of I/Os whose impact on TC is the most significant. However, identifying the minimal set of critical I/Os from a large pool of candidates is computationally complex so that heuristic-based approaches often struggle to optimize efficiently.

To tackle the challenge, this paper proposes a novel, explainable Graph Neural Network (GNN)-based framework. Our approach provides a fast and accurate method to identify the most critical uncontrollable I/Os for surgical TPI, maximizing TC recovery with the minimum possible overhead.

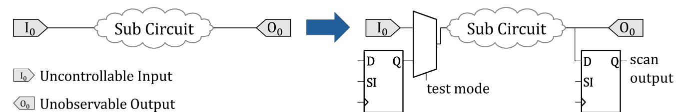

Fig. 1. Test Point Insertion technique for constrained inputs/outputs. Additional scan flip-flops are added to the masked input/output.

The main contributions of this paper are as follows:

- We propose a novel GNN-based framework that accurately predicts TC in sequential circuits with uncontrollable I/Os, providing the predictive engine for our efficient TPI strategy.
- We introduce an explainable AI (XAI) methodology that uses saliency maps to identify the most critical I/Os, enabling precise TPI to minimize area overhead while maximizing test coverage recovery.
- Our framework outperforms a leading DFT tool, achieving the target TC with 7.53% fewer TPs or improving TC by 4.34% for the same number of TPs on average.
- The effectiveness of the proposed framework is confirmed on large circuits and its technology independence is verified on an unseen PDK, proving its robustness and scalability for real-world applications.

II. Motivation and Related works

A. Test Coverage degradation under Uncontrollable I/Os

Uncontrollable I/Os, referred to in this paper as *masked I/Os*, can arise from various conditions in integrated circuits, including uninitialized sequential elements, black-box modules, analog-to-digital converters, or inaccessible sub-circuit interfaces [1]. These I/Os pose a significant challenge for Automatic Test Pattern Generation (ATPG) tools, often propagating unknown values that inhibit fault detection and lead to a drop in TC.

While comprehensive DFT solutions like boundary scan or IP wrapper chains can enforce full controllability, their significant area and performance overhead makes them prohibitively expensive for area-critical designs like memory peripheral circuits. For such applications, a more targeted approach is necessary.

TPI offers this targeted solution. As shown in Fig. 1, TPI is performed on critical I/Os to improve the controllability and

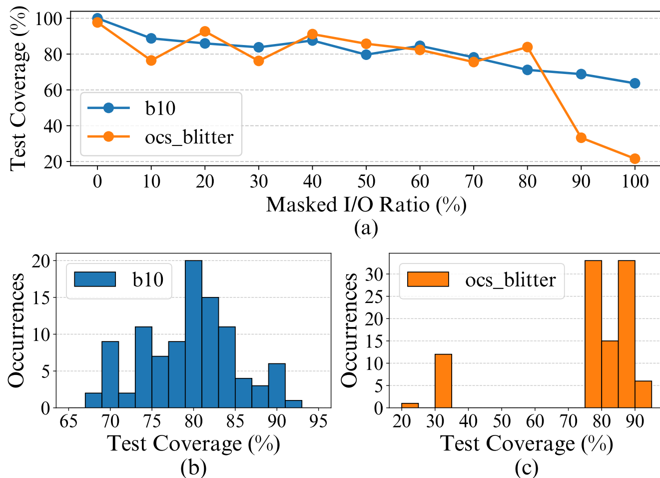

Fig. 2. Test coverage degradation on two fully scan-inserted circuits from ITC'99 [3] and OpenLane [4] under I/O masking. (a) Test coverage versus the percentage of randomly masked I/Os. (b) and (c) Distribution of test coverage over 100 trials, each with 50% of I/Os randomly masked

observability by inserting scan flip-flops. However, the resulting overhead in performance, area, and power consumption is a critical issue that necessitates a highly selective approach. Therefore, it is essential to identify the minimal set of TPs that would provide the most significant improvement in TC.

To quantify the impact of uncontrollable I/Os, we evaluate two benchmark circuits (`b10` and `ocs_blitter`) while masking various parts of I/O ports. Our analysis reveals that the impact of I/O masking on testability is both highly circuit-dependent and non-uniform. As shown in Fig. 2 (a), masking all I/Os results in a 40% TC reduction in `b10`, whereas there is an 80% TC drop in `ocs_blitter`. More importantly, for the same circuit, the specific selection of I/Os to mask plays a critical role. With 50% masking ratio, TC degradation for `ocs_blitter` ranges from negligible to over 70%, indicating that not all I/Os contribute equally to TC (Fig. 2 (c)). In some cases, masking a carefully selected 80% of I/Os leads to a smaller TC reduction than randomly masking 20%. These observations highlight the need for a careful strategy in selective TPI. However, identifying the optimal set of TPs is known to be NP-hard. As a result, commercial flows typically rely on heuristics or ATPG-guided search techniques to trade off coverage improvement against computational overhead.

B. Related Work

DFT on uncontrollable I/O: Numerous studies have focused on enhancing testability in circuits with uncontrollable I/Os particularly addressing the challenges posed by X-sources originating from inputs. For instance, the works in [1], [5], [6] propose efficient techniques for pattern generation and fault simulation to handle unknown values arising from uncontrolled inputs. However, these methods require time-consuming simulations, leading to excessively long execution times when applied to large-scale integrated circuits with numerous logic gates. Recent works [7], [8] introduce machine learning based approaches to predict testability in the presence of X-sources in a circuit and demonstrate that the impact of X-sources can be accurately and rapidly assessed without resorting to extensive simulations. While effective in predicting testability loss, they fail to provide a solution to alleviate the TC drop, which this paper aims to address.

ML for TPI: While machine learning (ML) and GNN-based approaches have recently advanced the field of Test Point Insertion (TPI), their scope has been largely confined to testability issues in the internal logic of the circuit, leaving controllability and observability limits at the I/Os unaddressed. These methods primarily focus on identifying internal nodes susceptible to Random-Pattern-Resistant (RPR) faults [9]–[11] or reducing test pattern counts [12], assuming that all I/Os in the circuit are fully controllable. Consequently, they are not designed to address TC degradation that originates from boundary constraints. To the best of our knowledge, our work is the first to formulate the TPI problem for uncontrollable I/Os, proposing a dedicated framework to efficiently alleviate the TC problem due to uncontrollable I/Os.

TABLE I
INITIAL NODE FEATURES FOR GNN TC PREDICTION.

Name	Dim.	Description	Related to
`logic_type`	5	One-hot vector representing cell type	Cell types
`fan_in`	1	Fan-in	Graph topology
`fan_out`	1	Fan-out	
`logic_depth`	1	The logic depth	Controllability
`dist_from_PO`	1	The distance from Primary Output	Observability
`masked`	1	1 if I/O port is masked, 0 otherwise	Testability
`SCOAP` [13]	6	SCOAP measures	
Total	**16**		

III. PROPOSED METHOD

A. Overview

We propose a novel method to insert test points on uncontrollable I/Os using explainable GNN. As illustrated in Fig. 3, the pipeline of the proposed method consists of three main stages: Data Generation, GNN-based TC prediction, and Saliency-based TPI. In the initial stage, the gate-level netlist is converted into AND-Inverter Graph (AIG) and a directed graph is constructed with generated features. A directed GNN is trained to predict TC for a circuit under various I/O masking configurations, avoiding simulation or ATPG feedback. After training, we compute a saliency map on the input features to quantify the marginal contribution of each I/O to TC. Based on this analysis, I/Os are ranked by importance, and boundary-only TPs are inserted at the top-ranked ports. Specifically, scan-mux control points capable of driving logic 0/1 are inserted at selected masked inputs, while scan observation flip-flops are inserted at selected masked outputs.

B. Data Generation

Before predicting TC with GNN, the gate-level netlist is flattened and converted into an AND-Inverter Graph (AIG) structure for technology-independent learning. To capture both testability and topological characteristics of the circuit graph, the initial features listed in Table I are selected.

The `logic_type` is a one-hot encoded vector that represents circuit components, including inputs, AND gates, inverters, latches, and outputs. The `fan_in` and `fan_out` capture

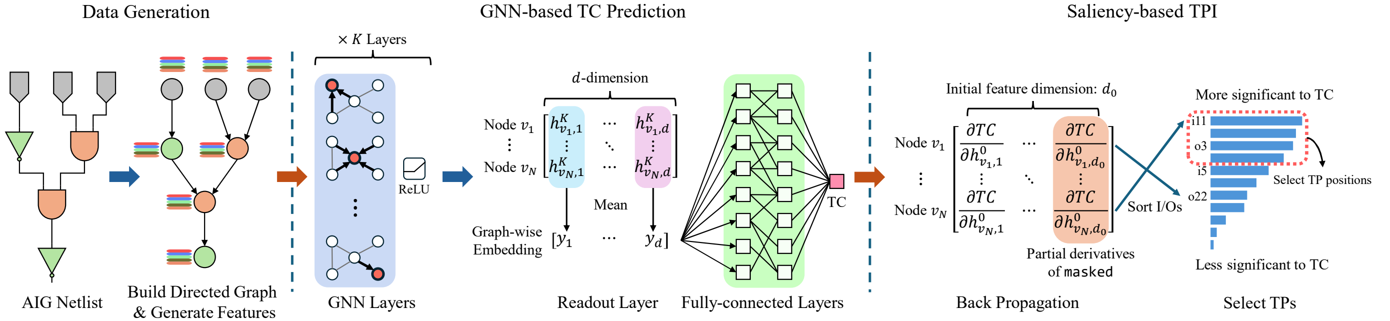

Fig. 3. The pipeline of the proposed method includes the following stages: *Data Generation*, which involves AIG conversion, directed graph construction, and feature generation; *GNN-based TC Prediction*, which processes data through GNN, Readout, and Fully connected (FC) layers; and *Saliency-based TPI*, which identifies test points with significant gradients impacting TC.

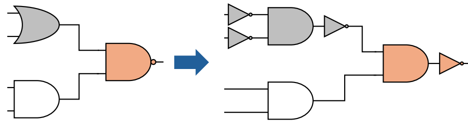

Fig. 4. An example of converting a netlist into AIG representation. OR gate and NAND gate in the original netlist are converted into AIG representation using only AND gates and NOT gates.

the graph topology and connectivity, while `logic_depth` and `dist_from_PO` provide indirect insights into circuit testability. The `masked` indicates whether the I/O port is uncontrollable, where 1 denotes an uncontrollable port and 0 represents a controllable port. To explicitly quantify testability, the Sandia controllability/observability analysis program (SCOAP) [13] is employed, and the resulting metrics are incorporated into the feature set. Each node is represented by a 16-dimensional feature vector, with all features normalized graph-wise to a range between 0 and 1.

The AIG format simplifies netlist representation by decomposing all logic gates into AND, inverter (NOT), and Latch elements. As illustrated in Fig. 4, gates such as OR and NAND from the original netlist are converted into equivalent AND-Inverter structures. Flip-flops (FFs) are transformed into latches, and input and output ports are represented as individual sets of nodes to efficiently handle uncontrollable I/Os. Converting a netlist to the AIG format offers two key advantages [14]: (i) it reduces the complexity associated with handling multiple types of logic gates by representing the entire netlist using only AND gates, NOT gates, and latches, thus simplifying the learning process for the GNN model. (ii) it achieves a technology-independent representation, which allows the proposed GNN to be applicable to netlist mapped to various technology libraries.

After converting the netlist to AIG format, the inputs, AND gates, and NOT gates are represented as vertices in the graph, with edges denoting the connections between the logic gates. It is important to highlight that the input graph for the GNN is directed, meaning the edges have a specific direction. This directionality is an essential aspect of graph construction, as it reflects the fault propagation flow, which is a fundamental consideration of DFT.

C. GNN-based Test Coverage Prediction

Using the graph data generated as described in Sec. III-B, GNN is employed to predict TC. The GNN model processes a graph $G = (V, E)$, representing the netlist as discussed in Sec. III-B, where each $v \in V$ is a logic gate and each $e \in E$ represents a directed connection between two gates. As depicted in Fig. 3, the GNN-based TC prediction model is composed of GNN layers, a Readout layer, and Fully-connected layers. After the GNN layers extract the low-dimensional representation (h_v^K) for each node, the Readout layer aggregates this information using graph-wise mean pooling, which averages features across all nodes. The pooled output is then passed through three Fully Connected layers, followed by a sigmoid activation function to generate the final TC prediction.

The proposed GNN-based framework offers a powerful data-driven approach by modeling the collective influence of circuit structure and node-level SCOAP metrics. This enables testability prediction through learned patterns directly from the circuit representation, eliminating the need for exhaustive simulation-based approaches utilized in commercial tools.

D. Saliency-based TPI Technique

To identify the optimal TP locations, we employ an Explainable AI (XAI) technique by generating a saliency map from the trained GNN. Originally developed for computer vision to identify pixels most influential to model predictions [15], a saliency map in our context highlights I/Os most critical to testability. The map is derived by calculating the gradient of the predicted test coverage ($\hat{TC}$) with respect to the initial input features of each node. While the full saliency map is a high-dimensional gradient, we focus on the derivative of the `masked` feature. Specifically, for each node v with d_0-dimensional initial features h_v^0, the saliency map $\mathcal{S}(h^0)$ is expressed as:

$$\mathcal{S}(h^0) = \nabla_{h^0}\hat{TC} = \begin{bmatrix} \frac{\partial \hat{TC}}{\partial h^0_{v_1,1}} & \cdots & \frac{\partial \hat{TC}}{\partial h^0_{v_1,d_0}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{TC}}{\partial h^0_{v_N,1}} & \cdots & \frac{\partial \hat{TC}}{\partial h^0_{v_N,d_0}} \end{bmatrix}. \quad (1)$$

Among the features, the gradient with respect to the `masked` feature, which we define as the 'Criticality' of each node (I/O) (Eq.2), quantifies the expected TC improvement if that I/O is made controllable.

$$\text{Criticality}(v) = -\frac{\partial \hat{TC}}{\partial \text{masked}_v} \quad (2)$$

Based on the criticality ranking, we iteratively insert TPs at the most critical I/Os – control points for inputs and observation points for outputs. After each iteration, the `masked` feature of the corresponding I/O is updated, and the saliency map is recomputed for the next iteration. This iterative and gradient-driven selection process avoids redundant TP insertions within the same logic cone, as the criticality of related nodes decreases following each iteration. Overall, the proposed technique provides an interpretable, scalable, and ATPG-free solution for test point insertion, enabling testability enhancement with minimal overhead.

IV. Experimental Results

A. Experimental Setup

To evaluate the proposed TPI method, the GNN is implemented using PyTorch framework and trained on an Intel Xeon Gold 6526Y CPU and an NVIDIA H100 NVL 94GB GPU. The proposed GNN is trained and tested using 43 circuits from the ISCAS'89 benchmark suite [16], ITC'99 benchmark suite [3], and OpenLane CI designs [4]. The circuits are synthesized using Design Compiler (DC) with Synopsys 32nm technology and converted to And-Inverter Graphs (AIG) using Yosys [17]. For each circuit, 11 I/O masking ratios are generated, ranging from 0% to 100% with an increase step size of 10%, resulting in a total of 473 samples. These samples are split into training, validation, and test sets in a 70:10:20 ratio. TMAX ATPG [18] is employed for TC (label) calculation and TC evaluation of both the proposed framework and the baseline. The labels for each circuit are normalized on a scale from 0 to 1, with 1.0 for TC without masking and 0.0 for TC with all I/Os masked.

During training, the GNN model parameters are optimized using the Adam optimizer with an initial learning rate of 0.001. The objective is to minimize the smoothed L1 loss between the predicted TC and the ground truth TC labels. Training is conducted for 200 epochs with a batch size of 8. We compare the proposed method against Synopsys TMAX Advisor [19] (i.e., baseline), which, to the best of our knowledge, is currently the only tool available in both academia and industry that supports TPI for uncontrollable I/Os.

B. TC Prediction

1) GNN Model Selection: To identify the most accurate predictor for our framework, we analyze the correlation between TC values generated by TMAX ATPG and those predicted by various GNN architectures, as summarized in Table II. The correlation analysis assesses how accurately the model captures changes in TC under different masked I/O configurations, compared to the ground truth TC from TMAX ATPG. Foundational models that are widely used for graph-based tasks (GCN, SAGE, and GAT), as well as DirGNN variants (DirGAT, DirGCN, and DirSAGE) [20], which are specifically designed to effectively model directed graph structures, are evaluated.

TABLE II
Correlation comparison between different GNN models

Model	Pearson Correlation		*Corr. Yield*
	Without SCOAP	With SCOAP	
DirGCN	0.731	**0.852**	*1.165*×
DirGAT	0.685	0.814	*1.188*×
DirSAGE	0.750	0.807	*1.076*×
GCN	0.813	0.771	*0.949*×
GAT	0.729	0.761	*1.044*×
SAGE	0.629	0.745	*1.184*×
Geo. Mean			*1.097*×

TABLE III
Statistics of circuits for evaluation

Benchmark	No. of Gates	No. of Ports	No. of Nodes	No. of Edges
chacha	11,196	80	61,038	91,020
PPU	10,843	105	43,250	64,981
s641	105	63	404	556
s38417	5,329	138	24,433	36,104
salsa20	15,548	81	84,977	126,308
synth_ram	6,258	96	34,009	50,935
usbf_device	15,719	183	70,222	105,661
wbqspiflash	1,559	108	6,707	10,115
zipdiv	767	111	3,988	5,840

As shown in Table II, DirGCN achieves the highest performance, with a Pearson correlation of 0.852, surpassing all other models. The key advantage of DirGNNs in this context lies in their ability to differentiate information flow based on edge direction. Unlike standard GNNs, which typically consider all neighbors uniformly, DirGNNs distinguish between predecessors (incoming edges) and successors (outgoing edges), processing their messages with distinct transformations (e.g., separate weight matrices). These directional messages are then aggregated, often by a weighted sum to form the final embedding of the node.

This explicit distinction mirrors fundamental testability concepts: controllability largely depends on forward signal propagation from predecessors, while observability relates to backward propagation towards successors. Based on these results and insights, DirGCN is chosen as the final model for subsequent experiments in our framework. The specific GNN model configuration used for the rest of the experiments we use 6 DirGCN layers with a hidden dimension of 64, and a 3-layer MLP head for generating the final TC output.

2) Impact of SCOAP Features: As discussed in Sec. III-B, Table I lists the features integrated as input to the proposed GNN model. Among these, SCOAP uniquely captures circuit testability characteristics through controllability and observability metrics. Therefore, to evaluate the impact of incorporating SCOAP features within the GNN framework for TC prediction, we conduct a comparison of correlation scores with and without SCOAP features.

Table II shows that the incorporation of SCOAP generally

TABLE IV
TP EFFICIENCY COMPARISON BETWEEN PROPOSED METHOD AND COMMERCIAL DFT TOOL

Benchmark	No. of TP Candidates*	Initial TC (%)	TMAX Advisor		Proposed Method			
			No. of TPs	*TP ratio***	No. of TPs	*TP ratio***	Saved TPs	TP efficiency over baseline***
chacha	74	83.52	45	*60.81%*	45	*60.81%*	0	0%
PPU	99	58.57	unreached	–	**12**	*12.12%*	–	uncomparable
s38417	134	79.36	3	*2.24%*	**2**	*1.49%*	1	0.75%
s641	59	30.54	59	*100.00%*	**53**	*89.83%*	6	10.17%
salsa20	75	90.98	37	*49.33%*	**23**	*30.67%*	14	18.66%
synth_ram	90	58.26	58	*64.44%*	**50**	*55.56%*	7	8.88%
usbf_device	177	79.84	45	*25.42%*	**20**	*11.30%*	25	14.12%
wbqspiflash	103	85.37	25	*24.27%*	25	*24.27%*	0	0%
zipdiv	105	90.38	42	*40.00%*	**34**	*32.38%*	8	7.62%
Average				*40.62%*		*31.23%*		7.53%

*Total test point insertable ports excluding control ports (i.e., clock, reset, scan_in ...) **Ratio of TP candidates to TPs
***TP efficiency is the difference of TP ratio between baseline and proposed method.

enhances the accuracy of the prediction in most models. For instance, DirGAT gives a correlation improvement of 1.188×. Similarly, DirGCN, which achieves the highest correlation with ground truth, shows an improvement of 1.165× in correlation when the SCOAP feature is included. These results clearly indicate that SCOAP enables the proposed framework to better capture circuit testability characteristics, leading to more accurate TC predictions.

C. Comparison with Commercial TPI tool

We compare the proposed method with the baseline on 9 unseen benchmarks. The statistics of the evaluated circuits are listed in Table III. Table IV presents a detailed comparison of TPI performance between the proposed method and the commercial TPI tool, TMAX Advisor [19], hereafter referred to as the *baseline*. In the second column, TP Candidate refers to the number of I/Os that are TP insertable, excluding control and scan-related pins (e.g., clk, reset, scan_in, and scan_out). Initial TC refers to the TC with all TP candidates masked, excluding control pins and scan-related pins. For each benchmark, we report the minimum number of TPs (No. of TPs) and the corresponding TP ratio (inserted TPs / TP candidates) required to reach 95% TC. TP efficiency denotes the difference in TP ratio between the baseline and the proposed method for achieving this 95% TC.

As shown in Table IV, our framework demonstrates superior TP efficiency compared to the baseline TMAX Advisor. On average, our method achieves the 95% target TC using 7.53% fewer TPs. For instance, in `salsa20`, the proposed method achieves target TC by inserting TPs on 30.67% of I/Os (23 TPs inserted out of 75 TP candidates) to achieve the target TC, while the baseline tool required 49.33% of I/Os (37 TPs inserted out of 75 TP candidates). On `PPU`, our method reaches 95% TC with just 12 test points, whereas TMAX Advisor fails to achieve the target TC of 95% after inserting until its abort limit.

To further evaluate performance, we compare the final TC achieved by both methods using an equal TP budget. For this comparison, the number of inserted TPs is fixed to match the count required by our method to reach a 95% TC target. As shown in Table V, our framework delivers an average TC improvement of 4.34%, with gains reaching up to 13% on some benchmarks. This improvement is particularly noteworthy given that the baseline already achieves high TC. These additional gains demonstrate that our method is highly effective at identifying the most critical I/Os which contribute to the final, hard-to-cover faults, thereby pushing TC beyond the conventional methods.

TABLE V
COMPARISON OF TC WITH SAME NUMBER OF TPS

Benchmark	No. of TPs	TC (%)		TC Improvement
		TMAX Advisor	Proposed Work	
chacha	45	95.13	95.10	-0.03
PPU	12	82.71	95.82	13.11
s38417	2	94.85	95.34	0.49
s641	53	92.50	95.54	3.04
salsa20	23	94.59	95.01	0.42
synth_ram	50	83.16	95.07	11.91
usbf_device	20	90.08	98.79	8.71
wbqspiflash	25	96.75	96.61	-0.14
zipdiv	34	94.38	95.89	1.51
Average		91.27	95.91	4.34

TABLE VI
STATISTICS OF LARGE CIRCUITS

Benchmark	No. of Gates	No. of Ports	No. of Nodes	No. of Edges
FFT256	124.4K	92	714.9K	1,071.9K
NVDLA	108.8K	27,595	1,367.0K	1,966.0K
gfx_top	176.1K	294	1,126.4K	1,699.6K

D. Towards scalable and PDK independence

To demonstrate both scalability and technology independence, we evaluate the proposed framework on three large open-source circuits (>100K gates, detailed in Table VI) [21]–[23], synthesized using a TSMC 28nm technology. TPs are inserted to reach a target TC of 95% except for NVDLA. It should be noted that the GNN model is trained using circuits synthesized with a 32nm technology node, allowing us to evaluate the PDK independence in the proposed work.

For NVDLA, since TMAX Advisor reaches an insertion limit of 3996 TPs, we report the TC (93.31%) for the same number of TPs in the proposed method to ensure a fair comparison. As shown in Table VII, the proposed method efficiently identifies critical I/Os for TP insertion, demonstrating an average TP efficiency of 4.72%. This result highlights the effectiveness of the GNN-based saliency guidance in navigating the large search space inherent in these complex designs. The strong performance of the proposed framework

TABLE VII
EXPERIMENTAL RESULTS ON LARGE CIRCUITS WITH DIFFERENT PDK

Benchmark	No. of TP Candidates	Initial TC (%)	TMAX Advisor			Proposed Work				
			No. of TPs	*TP ratio*	TC** (%)	No. of TPs	*TP ratio*	TC** (%)	Saved TP	TP efficiency***
FFT256	65	62.02	2	*3.08%*	95.24	4	*6.15%*	95.00	-2	-3.07%
NVDLA*	27,568	81.02	3,996	*14.50%*	93.31	3996	*14.50%*	93.31	0	0%
gfx_top	267	85.82	116	*43.46%*	99.55	70	*26.22%*	98.80	46	17.24%
Average				*20.35%*			*15.62%*			4.72%

*NV_NVDLA_CMAC_CORE_active **First TC exceeding 95% as TP increases
***TP efficiency is the difference of TP ratio between baseline and proposed method.

on these large circuits (>100k gates) synthesized on unseen technology node, validates its scalability and the robustness of the technology-independent AIG-based approach for practical, large-scale TPI tasks.

TABLE VIII
RUNTIME COMPARISON WITH COMMERCIAL DFT TOOL

Benchmark	No. of TPs	TMAX Advisor	Proposed Method	
		Runtime (sec)	Runtime (sec)	Runtime per TP (sec)
chacha	74	0.84	1.13	0.015
PPU	84	0.59	1.38	0.016
s38417	86	0.07	3.58	0.042
s641	59	0.02	1.79	0.030
salsa20	76	0.97	2.70	0.036
synth_ram	59	1.31	1.58	0.027
usbf_device	71	0.32	6.14	0.087
wbqspiflash	68	0.06	1.15	0.017
zipdiv	68	0.04	1.17	0.017
FFT256	26	4.69	**2.56**	0.099
NVDLA	3,996	50.93	704.25	0.176
gfx_top	116	17.23	**15.77**	0.136

E. Runtime Comparison

Table VIII presents the runtime comparisons between the baseline tool and the proposed method. For the proposed method, the number of inferences directly corresponds to the number of TPs inserted. To ensure a fair comparison, the number of iterations for the proposed method is limited to the maximum number of TPs inserted by the baseline tool for each respective circuit. Both the full iterative TPI process and per-TP analysis for the proposed method are shown in the Table VIII.

For smaller benchmark circuits, the iterative proposed method shows a slightly longer runtime than the baseline. However, its absolute execution time is generally less than 10 seconds, making this difference practically insignificant. More importantly, for larger circuits (>100k gates), a clear trend is observed that the proposed method demonstrates superior scalability. Except for the NVDLA benchmark, the proposed framework operates, on average, 1.46× faster on large circuits than the baseline. While the runtime of the baseline tool increases substantially with circuit size, our GNN-based framework leverages GPU parallelism, resulting in much slower growth in the execution time.

The relatively long runtime in the NVDLA benchmark can be explained by its unusually high I/O count relative to its gate count (>27k ports for 108k gates), resulting in a low gate-to-port ratio. This structural characteristic is arguably atypical for a single large design. For more common large designs (>100k gates) exhibiting higher gate-to-port ratios (i.e., greater internal complexity relative to boundary pins), the baseline runtime is expected to increase even more sharply, while the GPU-bound inference time of the proposed method remains relatively stable. Therefore, it should be noted that the speedups observed on other large benchmarks such as `FFT256` and `gfx_top` are likely more representative of the performance gains expected for typical complex commercial designs. In summary, the proposed method offers acceptable runtime for smaller circuits and demonstrates substantial, scalable runtime advantages for large, high-complexity designs where traditional methods become bottlenecks.

V. CONCLUSIONS

This paper presents a novel, explainable GNN-based framework for TPI that predicts TC of digital circuits with uncontrollable I/Os. To address the limitations of traditional simulation-based TPI methods, the proposed method utilizes GNN framework to predict TC by incorporating quantitative SCOAP metrics as initial node features, thereby capturing the combined impact of circuit topology and inherent testability. Saliency map analysis (XAI) technique is then applied to identify and prioritize critical I/Os for TPI, enhancing interpretability.

The experimental results indicate the effectiveness of the proposed framework. Compared to a commercial DFT tool (baseline), the proposed method achieves a target TC of 95% with 7.53% fewer TPs on average. When comparing performance at an equal number of inserted TPs, the proposed TPI method yields an average TC improvement of 4.34% over the baseline. Furthermore, the proposed framework demonstrates robust scalability on large circuits (>100k gates) and technology independence, validated through an AIG representation and evaluation across different process technologies (28nm and 32nm). In summary, this work integrates GNN-based prediction with XAI-driven interpretation for TPI, offering a fast, scalable, interpretable, and effective data-driven optimization approach to enhance circuit testability with uncontrollable I/Os.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.RS-2024-00405495, Plug&Play(P&P) Chiplet Integration research center), in part by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.RS-2025-09942968, AI Semiconductor Innovation Lab(Yonsei University)), and in part by Samsung Electronics. The EDA Tools used in this work were supported by IDEC, Daejeon, South Korea.

REFERENCES

[1] S. Hillebrecht, M. A. Kochte, D. Erb, H.-J. Wunderlich, and B. Becker, "Accurate qbf-based test pattern generation in presence of unknown values," in 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2013, pp. 436–441.

[2] M. Songhorzadeh and R. N. Asli, "A low area overhead ieee-1500-compliant wrapper for embedded memories," International Journal of Computer Theory and Engineering, vol. 4, no. 6, p. 944, 2012.

[3] F. Corno, M. S. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," IEEE Design & Test of computers, vol. 17, no. 3, pp. 44–53, 2000.

[4] A. Ghazy and M. Shalan, "Openlane: The open-source digital asic implementation flow," in Proc. Workshop on Open-Source EDA Technol.(WOSET), 2020.

[5] S. J. Chandra and J. H. Patel, "Accurate logic simulation in the presence of unknowns." in ICCAD, 1989, pp. 34–37.

[6] K. Scheibler, D. Erb, and B. Becker, "Accurate cegar-based atpg in presence of unknown values for large industrial designs," in 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2016, pp. 972–977.

[7] M. Pradhan, B. B. Bhattacharya, K. Chakrabarty, and B. B. Bhattacharya, "Predicting X-Sensitivity of circuit-inputs on test-coverage: a machine-learning approach," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 12, pp. 2343–2356, 2018.

[8] T.-H. Nguyen, Y. Ju, D. Yoon, and H. Choi, "Learning gate-level netlist testability in the presence of unknowns through graph neural networks," in Proceedings of the 30th Asia and South Pacific Design Automation Conference, 2025, pp. 622–627.

[9] Y. Ma, H. Ren, B. Khailany, H. Sikka, L. Luo, K. Natarajan, and B. Yu, "High performance graph convolutional networks with applications in testability analysis," in Proceedings of the 56th Annual Design Automation Conference 2019, 2019, pp. 1–6.

[10] Z. Shi, M. Li, S. Khan, L. Wang, N. Wang, Y. Huang, and Q. Xu, "Deeptpi: Test point insertion with deep reinforcement learning," in 2022 IEEE International Test Conference (ITC). IEEE, 2022, pp. 194–203.

[11] Z. Chao, B. Sun, H. Lyu, G. Yu, M. Wang, W. Li, Z. Liu, J. Mu, S. Liang, J. Ye et al., "Hightpi: A hierarchical graph based intelligent method for test point insertion," in 2025 IEEE 43rd VLSI Test Symposium (VTS). IEEE, 2025, pp. 1–7.

[12] B. Bhaskaran, S. Banerjee, K. Narayanun, S.-C. Hung, S. N. M. Mojaveri, M. Liu, G. Chen, and T.-C. Liang, "Observation point insertion using deep learning," in Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design, 2022, pp. 1–8.

[13] L. H. Goldstein and E. L. Thigpen, "Scoap: Sandia controllability/observability analysis program," in Proceedings of the 17th Design Automation Conference, 1980, pp. 190–196.

[14] M. Li, S. Khan, Z. Shi, N. Wang, H. Yu, and Q. Xu, "Deepgate: Learning neural representations of logic gates," in Proceedings of the 59th ACM/IEEE Design Automation Conference, 2022, pp. 667–672.

[15] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2014. [Online]. Available: https://arxiv.org/abs/1312.6034

[16] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in 1989 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 1989, pp. 1929–1934.

[17] C. Wolf, "Yosys open synthesis suite," https://yosyshq.net/yosys/.

[18] Synopsys. (2025) Testmax atpg. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/test-automation/testmax-atpg.html

[19] Synopsys. (2025) Testmax advisor built on synopsys spyglass. [Online]. Available: https://www.synopsys.com/content/dam/synopsys/implementation%26signoff/datasheets/testmax-advisor-ds.pdf

[20] E. Rossi, B. Charpentier, F. Di Giovanni, F. Frasca, S. Günnemann, and M. M. Bronstein, "Edge directionality improves learning on heterophilic graphs," in Learning on Graphs Conference. PMLR, 2024, pp. 25–1.

[21] OpenCores Community, "Pipelined fft/ifft 256 points processor," 2010, created: Feb. 1, 2010, Updated: Jul. 30, 2014, Accessed: Apr. 22, 2025. [Online]. Available: https://opencores.org/projects/pipelined_fft_256

[22] NVIDIA Corporation, "Nvidia deep learning accelerator (nvdla)," 2018, accessed: Apr. 22, 2025. [Online]. Available: https://nvdla.org/

[23] OpenCores Community, "Orsoc graphics accelerator," 2012, created: Mar. 22, 2012, Updated: Jul. 2, 2017, Accessed: Apr. 22, 2025. [Online]. Available: https://opencores.org/projects/orsoc_graphics_accelerator