Transferable Graph Neural Network-based Delay-Fault Localization for Monolithic 3D ICs*

Shao-Chun Hung, Sanmitra Banerjee, Arjun Chaudhuri, Jinwoo Kim, Sung Kyu Lim, *Senior Member, IEEE,* and Krishnendu Chakrabarty, *Fellow, IEEE*

***Abstract*—Monolithic 3D (M3D) integration is a promising technology for achieving high performance and low power consumption. However, the limitations of current M3D fabrication flows lead to performance degradation of devices in the top tier and unreliable interconnects between tiers. Fault localization at the tier level is therefore necessary to enhance yield learning, For example, tier-level localization can enable targeted diagnosis and process optimization efforts. In this paper, we develop a graph neural network-based diagnosis framework to efficiently localize faults to a device tier. The proposed framework can be used to provide rapid feedback to the foundry and help enhance the quality of diagnosis reports generated by commercial tools. Results for four M3D benchmarks, with and without response compaction, show that the proposed solution achieves up to 32.86% improvement in diagnostic resolution with less than 1% loss of accuracy, compared to results from commercial tools. The proposed framework has also been demonstrated to be transferable to perform diagnosis on various design configurations without performance degradation.**

***Index Terms*—Monolithic 3D integration, Graph neural network, Diagnosis**

I. Introduction

As Moore's law reaches physical limits, three-dimensional (3D) integration is now being adopted for integrated circuits (ICs). In today's 3D technology, die/wafer bonding with through-silicon vias (TSVs) is being used due to its minimal impact on current fabrication flows. However, keep-out-zones around TSVs (necessary to prevent wire damage due to tensile stress) can create routing blockages and increase the chip footprint and total wirelength. Monolithic 3D (M3D) integration has emerged as a promising technology to achieve higher performance and lower power consumption compared to 2D and die/wafer bonded 3D ICs [1]. M3D leverages fine-grained monolithic inter-tier vias (MIVs) to achieve high-precision alignment and extremely thin device layers [2]. The size of MIVs is of the same order of magnitude as conventional back-end-of-line (BEOL) vias. As a result, a large number of MIVs can be used in M3D designs, leading to a significant reduction in wirelength.

Despite these advantages, M3D introduces several challenges that must be addressed before this technology can be widely adopted. Temperature management during fabrication is one of the major concerns. Typically, thermal budgets of transistor manufacturing processes exceed 1000°C (e.g., for dopant activation) [3]. However, in M3D designs, the fabrication of upper-tier transistors in M3D designs with typical thermal budgets causes damage to wires and cells underneath [4]. While advanced processes have been developed to fabricate transistors at a low temperature, they can cause up to 20% performance mismatch between the devices in different tiers [5]. The reliability of interconnects is another concern for M3D ICs. Standard copper/low-k BEOL cannot be used between tiers because the fabrication steps in the upper tiers pose contamination risks, while low-k dielectrics are thermally unstable after annealing processes [6]. As an alternative, tungsten has good thermal stability, but its intrinsic resistance is six times larger than copper, leading to an increase in RC delay in the lower tiers [7]. Moreover, MIVs in M3D designs are prone to defects as they penetrate through the inter-tier dielectric. Surface roughness can produce voids in the dielectric [8], which may lead to voids in MIVs during etching, resulting in delay defects and degradation of circuit performance [9]. These defects due to immature manufacturing processes tend to be manifested as systematic delay faults that are located in the same tier. It is necessary to prevent such fabrication-related defects before M3D can become ready for commercial exploitation. Delay-fault diagnosis is therefore important in order to provide early feedback to the foundry and facilitate yield learning.

In contrast to die/wafer bonding in stacked 3D integration, tiers in M3D designs are fabricated *in situ*, which makes it hard to ensure a known-good tier before assembly. Post-assembly methods such as [10] are not applicable to M3D due to the large area overhead for wrapper cells around MIVs. In addition, delay-fault diagnosis catered to M3D designs is especially important as tiers in M3D ICs suffer from different systematic defects due to immature manufacturing processes. Such tier-specific fabrication-related defects do not exist in 2D designs; therefore, they are overlooked by previous work. [11] leverages unsupervised learning to extract good candidates in diagnosis reports. However, the extracted candidates can be located in different tiers in M3D designs, which is not sufficient to provide the high level of resolution (i.e., fault localization) needed at the tier level. [12] proposed a built-in-self-test (BIST) solution for MIV testing and diagnosis. However, the BIST structure does not localize faults to a specific tier and requires dedicated test tiers between each pair of device tiers, which increases the manufacturing cost. In [13], an observation-point insertion algorithm was developed for tier-level fault localization, but the impact of this solution on fault diagnosis was not studied and the area overhead becomes prohibitive for the likely scenario of a large number of MIVs. To make M3D integration feasible, there is a need for a diagnosis framework that can efficiently localize faults to a tier. Such a diagnosis framework should provide early feedback to the foundry before the time-consuming physical failure analysis (PFA). For example, an immature manufacturing process can

*This research was supported in part by the National Science Foundation under grant CCF-1908045. A preliminary version of this paper was published in *Design, Automation Test in Europe Conference Exhibition*, 2022

result in a large number of chips failing on the tester with defects located in the same tier. Tier-level fault localization makes it possible for the foundry to review its processes for the predicted faulty tier without waiting for further analysis; therefore, yield learning is accelerated. An effective diagnosis method should also be compatible with existing diagnosis flows provided by commercial tools to improve the quality of diagnosis.

In this paper, we propose a novel machine learning-based (ML-based) diagnosis framework for M3D ICs to locate faults at the tier level. We focus on at-speed transition delay fault (TDF) diagnosis because the M3D-specific defects discussed above tend to be manifested in the form of delay faults that impact circuit timing. Our method is able to localize faults based on the circuit netlist and failure log files from the tester. The key contributions of this paper are as follows:

- We develop two models, *Tier-predictor* and *MIV-pinpointer*, based on graph neural networks (GNNs) to locate faults at the tier level and in MIVs.
- We develop a GNN-based policy to improve the quality of diagnosis reports.
- We ensure the compatibility of the proposed method with conventional scan-based designs and commercial tools, both with and without test compression.
- We show that the proposed framework is transferable such that diagnosis can be carried out for designs with various design configurations.
- We demonstrate that the proposed framework can be synergistically combined with previous work to provide the high level of resolution at the tier level.
- The proposed framework simply utilizes the circuit netlist and failure log files from the tester for making predictions; therefore, test cost is minimized as no additional test time is needed to generate diagnostic data.

The rest of the paper is organized as follows. Section II provides an overview of M3D integration, logic diagnosis, and GNN. Section III presents the proposed diagnosis framework. In Section IV, we conduct transferability analysis and provide our solutions to improve the transferability of the GNN-based framework. Section V presents the proposed candidate pruning and reordering algorithm. We compare the effectiveness of our framework with a commercial fault-diagnosis tool in Section VI. In Section VII, we discuss the transferability of our framework between designs and provide guidance on choosing appropriate models for diagnosis. Finally, Section VIII concludes the paper.

II. Background

A. Monolithic 3D Integration

M3D integration processes active device tiers sequentially on a single wafer. M3D integration has the potential to enable a wide variety of applications. M3D NAND flash memory has been commercially produced in recent years due to better performance and lower cost compared to 2D planar NAND Flash [14]. In [15], an M3D nonvolatile random-access memory (NVRAM) was proposed for AI accelerators. The 3D-integrated interface helped in the alleviation of memory-bounded problems, both during training and inference. In [16], heterogeneous M3D systems, i.e., multiple technology nodes for different tiers, were predicted to be promising solutions for next-generation wireless communication.

Research efforts are also being devoted to M3D testing and diagnosis. [13] developed an observation-point insertion algorithm for tier-level fault localization. A test pattern reshaping algorithm was proposed in [17] to reduce PSN-induced voltage droop during M3D delay testing. However, fault diagnosis for M3D-specific defects has not been addressed in prior work. This is critical because tiers in an M3D design suffer from different fabrication-related limitations and process variations. For example, defects arising from the relatively immature low-temperature processes and the bonding interface of inter-layer dielectric and upper tier's active layer typically influence transistors in the upper tiers, while delay faults due to unreliable interconnects between tiers affect the timing in the bottom tiers [5] [6]. Tier-level diagnosis is thus important to localize faults to a tier, enabling efficient PFA and technology bringup.

B. Logic diagnosis

Logic diagnosis is used to identify potential defect locations when a chip fails on the tester. A diagnosis process aims to provide an accurate guide to the subsequent PFA step. Three important measures are used to evaluate the quality of a diagnosis algorithm: (i) diagnostic resolution, (ii) accuracy, and (iii) first-hit index (FHI) [18]. Diagnostic resolution is defined as the number of fault candidates in a diagnosis report; accuracy is determined by whether one of the candidates pinpoints the ground-truth defect location. Ideally, the diagnostic resolution should be 1, but it is hard to ensure that the only identified candidate is the ground-truth defect location. An efficient diagnosis methodology needs to find a trade-off between resolution and accuracy. A diagnosis report is ranked with the most probable candidate listed at the top. FHI refers to the index of the first candidate that is actually a ground-truth defect location. Smaller the FHI, better the diagnosis process.

Test compression is widely used in modern IC designs to achieve a significant reduction in test time and data volume; however, the test-compression environment increases the difficulty of identifying the ground-truth defect locations during diagnosis. In the proposed framework, we aim at improving diagnostic resolution for M3D designs, both with and without response compaction. Our tier-level predictions are used to enhance the quality of diagnosis reports generated by an automatic test pattern generation (ATPG) tool. This is a key benefit of the proposed solution−it is synergistic and compatible with commercial tools. In addition, ML-aided MIV diagnosis can help in the early characterization of defective MIVs.

C. Graph Neural Network (GNN)

GNN is an ML method that processes data on graphs. In the field of IC design, GNN has attracted special attention because it can carry out computations directly in non-Euclidean domains. ML models such as recurrent neural networks and convolutional neural networks are not effective for graph-structured data because they operate on Euclidean

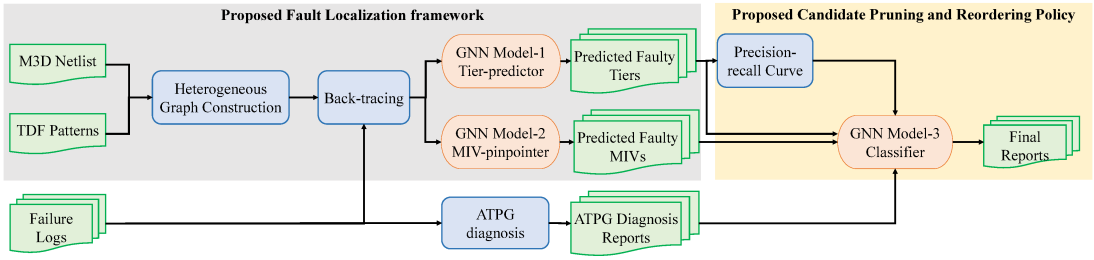


Fig. 1: GNN-based fault diagnosis flow.

data such as images and text sequences. However, different graphs have different numbers of nodes/edges and irregular node connections. A pre-processing phase is therefore required to map graph structures to simplified representations, due to which topological dependency of each node may be lost during pre-processing [19].

GNNs have been applied to solve different types of IC design problems in recent years. In [20], a GNN-based model for distributed circuit designs is developed. The proposed model is able to replace the time-consuming eletromigration simulators to accurately predict the electromigration properties. [21] leverages the graph attention network to help estimate individual net length within a circuit before placement. Furthermore, ML is well suited to IC diagnosis because a large volume of data is collected throughout the production and product lifetime [22]. This advantage and the effectiveness of GNN motivate us to design a GNN-based framework for tier-level diagnosis. For our diagnosis problem, GNN models can learn the complex, non-linear relationship between a fault location (root-cause) and the failure response (effect). The trained models can then be used to directly predict the faulty tier and MIVs by providing only the failure log from the tester as input. This is significantly faster than running fault simulation for each candidate fault and matching the failure response with what we got from the tester. Therefore, the proposed solution can provide feedback to the foundry and improve ATPG diagnosis reports without runtime overhead.

The transferability of ML is an important property that enables pre-trained models to be applicable to new data without retraining [23]. Transfer learning has been shown to be effective on graph-structured data and GNNs [24] [25]. In the field of IC diagnosis, the circuit under diagnosis (CUD) can be synthesized with different configurations. Furthermore, various partitioning methods have been developed for M3D designs to partition standard cell gates into device tiers [26] [27]. Collecting data and training new models for each CUD require additional runtime and computational efforts, which negates the benefits of leveraging ML models in the diagnosis process. This motivates us to develop a transferable framework that can be directly applied to new circuits without retraining. An efficient framework should accurately localize faults to a device tier and help improve the quality of diagnosis reports for CUDs with different design configurations.

III. Proposed Fault Localization Framework

In this section, we describe our GNN-based framework for tier-level fault diagnosis in M3D ICs. Fig. 1 presents a flowchart for the proposed diagnosis method. The first step is to convert the CUD into a graph object. Next, given a failure log file from the tester, we simultaneously generate our GNN-based predictions and launch the ATPG diagnosis process. Finally, we utilize the prediction results to reorder and prune candidates from the ATPG diagnosis report to generate the final candidate list. Our framework is implemented in PyTorch with the Deep Graph Library (DGL) package [28].

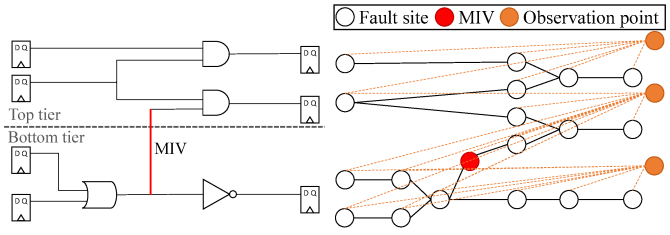
TABLE I: Features in a heterogeneous graph.

Symbol	Granularity	Object	Description
N_{fi}	Circuit-level	Node	Number of fan-in edges
N_{fo}	Circuit-level	Node	Number of fan-out edges
T_{pat}	Circuit-level	Node	Transitions with TDF patterns
N_{top}	Circuit-level	Node	Number of fan-in Topedges
Loc	Circuit-level	Node	Tier-level location
Lvl	Circuit-level	Node	Level in topological order
Out	Circuit-level	Node	Whether it is a gate output
MIV	Circuit-level	Node	Whether it connects to an MIV
D_{top}	Top-level	Edge	Shortest distance between both ends
N_{MIV}	Top-level	Edge	Number of MIVs passed through

A. Heterogeneous Graph Structure

The first step in our framework is to transfer a CUD into a heterogeneous graph, which incorporates different types of nodes and links in the graph structure. There are two levels in the heterogeneous graph. At the circuit level, the CUD is converted to a graph, where each fault site (i.e., every pin of a gate) forms a node, while edges are composed of input-pin-to-output-pin and net-stem-to-net-branch connections. In addition to fault sites, we also represent each MIV as a node in the graph. This is important because MIVs are prone to delay defects in M3D designs (see Section I). However, conventional TDF testing does not provide such fine-grained resolution. A post-processing step is required in conventional TDF testing to evaluate whether there is an MIV between a top-tier gate and a bottom-tier gate and whether such an MIV is faulty. Given a CUD with n gates, the time complexity of this step is $\mathcal{O}(n^2)$. By adding MIV nodes in the proposed graph structure, each MIV can be pinpointed in constant time.

Next, we construct nodes and edges at the top level of the CUD, denoted as *Topnodes* and *Topedges*, respectively, to complete the heterogeneous graph structure. A Topnode corresponds to an observation point (i.e., the input of a scan flop) during scan testing. Each Topnode is connected to all the nodes in its fan-in cone by Topedges. Fig. 2 illustrates the construction of our graph structure from a CUD. After graph construction, we apply ATPG patterns and conduct simulation with multiple logic values [29] to memorize transitions (i.e.,



(a) Circuit netlist (b) Heterogeneous graph

Fig. 2: Illustration of the proposed heterogeneous graph structure.

Input: Heterogeneous graph G, failure log file f from the tester
Output: sub-graph $G' \in G$

```
1  V' := FaultSites(G)
2  foreach erroneous test output response r in f do
3      p ← FailedPattern(r), T ← FailedTopnode(r), N := ∅
4      foreach Topnode n in T do
5          foreach fault site s in Successors(n) do
6              if s has a transition state with p then
7                  N := N ∪ s
8              end
9          end
10     end
11     V' := V' ∩ N
12 end
13 G' := CreateSubgraph(G, V')
14 return G';
```

Fig. 3: Pseudo-code for the back-tracing algorithm.

whether a node switches from $0(1)$ to $1(0)$). We also find the shortest path between both ends of a Topedge. The number of nodes and the number of MIVs in such a shortest path establish the Topedge features. Details of features in a heterogeneous graph are shown in Table I. Note that to convert a CUD to a graph and memorize transitions with TDF patterns, every gate and net is explored, so the time complexity is $\mathcal{O}(|V| + |E|)$, where $|V|$ is the number of nodes at the circuit-level, and $|E|$ is the number of edges at the circuit-level. Moreover, we leverage breadth-first search (BFS) to collect the fan-in cone of each Topnode during the top-level graph construction. Because our circuit-level graph is unweighted, paths found by BFS between the source node and traverse nodes are guaranteed to be the shortest. Hence, Topedge features can be calculated simultaneously during the graph construction, where the time complexity is $\mathcal{O}(|V| + |E|)$ [30]. Other features listed in Table I are built in constant time. Therefore, the overall time complexity for the heterogeneous graph construction is $\mathcal{O}(|V| + |E|)$.

The top-level graph strengthens the relationships between observation points and their fan-in nodes; this is important for logic diagnosis because only the fan-in nodes can be the candidate fault locations when observation points capture erroneous responses. Although the generation of Topnodes and Topedges requires additional runtime and memory, it needs to be run only once for each benchmark and can be reused for every failure log file; therefore, the runtime and memory overhead are not concerns and the cost is easily amortized.

TABLE II: Initial node features in a sub-graph. Length of a Topedge is the shortest distance between its source and destination nodes.

Description	Type	Significance score
Number of fan-in edges in the circuit	Numerical	0.4959
Number of fanout edges in the circuit	Numerical	0.4916
Number of Topedges connected	Numerical	0.4995
Tier-level location	Binary	0.4957
Level in topological order	Numerical	0.4996
Whether it is a gate output	Binary	0.4919
Whether it connects to an MIV	Binary	0.4904
Number of fan-in edges in the sub-graph	Numerical	0.4958
Number of fanout edges in the sub-graph	Numerical	0.4991
Mean length of Topedges connected	Numerical	0.4926
Standard deviation of length of Topedges connected	Numerical	0.4923
Mean number of MIVs passed through by Topedges connected	Numerical	0.4959
Standard deviation of number of MIVs passed through by Topedges connected	Numerical	0.4887

B. Back-tracing

Fig. 3 sketches the steps involved in back-tracing. Lines 2-12 iterate through every erroneous output response. Line 3 finds the pattern p with which the current response is observed on the tester and collects a set T of Topnodes that connect to the test output where the response is captured. Lines 4-10 iterate through all the nodes in the input cones of Topnodes in T. Note that only nodes whose signal values switch during scan capture when p is applied are capable of activating delay faults and producing an erroneous response. Therefore, Line 7 collects the union of such nodes to form a suspect list corresponding to the current response. In Line 11, the intersection of suspect lists for every erroneous response becomes the final candidate list for the input failure log file. Finally, Line 13 extracts all nodes in the candidate list to generate a sub-graph for the subsequent GNN models. Note that the top level in the proposed heterogeneous graph is solely used to accelerate the back-tracing process. After back-tracing, only nodes at the circuit level are extracted to create a homogeneous sub-graph. The topological dependency at the top level is encoded as numerical features of the extracted sub-graph.

The time complexity of the above back-tracing procedure can be analyzed as follows. Given a failure log file with n_r erroneous responses, Lines 2-12 are executed n_r times to find the candidate list. Let the number of gates in the graph G be n_G. During the evaluation in Lines 4-10, each node in the fan-in cone is analyzed at most $|T|$ times, where $|T|$ is a constant referred to as the number of Topnodes connected to an output channel; hence the time complexity is $\mathcal{O}(n_G)$. In Line 13, the time complexity of finding the intersection of two subsets of G is $\mathcal{O}(n_G)$. The other steps are completed in constant time. Therefore, the overall time complexity is $\mathcal{O}(n_r n_G)$.

C. Proposed GNN Models

We utilize the graph convolutional network (GCN) [31] to train our Tier-predictor and MIV-pinpointer. Sub-graphs generated after the back-tracing step are fed into the GCN models, with the initial node features listed in Table II. Features at the circuit level are extracted based on the topological dependency

in the CUD, which has been shown to be effective in solving EDA problems using GNNs [21]. Moreover, we encode the connections at the top level (e.g., the length of Topedges and the number of MIVs passed through by Topedges) as numerical features to represent the relationships between each node and scan outputs. To demonstrate the importance of the selected features, we leverage the GNNExplainer [32] to identify the significance of each feature to the classification labels. Significance scores are shown in Table II. A feature is more important for the classification task if its significance score is closer to 1. Clearly, features at the top level are of the same importance as features at the circuit level. This is because the length of Topedges and the number of MIVs passed through by Topedges demonstrate how difficult each fault effect can be captured by scan outputs and whether the propagation paths are prone to MIV defects, respectively. Such features are key factors that affect the testing and diagnosis processes, where failure logs are generated. As both features at the circuit level and features at the top level contribute to the classification task, we utilize all the selected features to train our GCN models.

To gather information and learn from neighbors of a node n, GCN layers are added to aggregate its features as follows [31]:

$$h_n^{(l+1)} = \sigma\left(b^{(l)} + \Sigma_{i\in\mathcal{N}(n)} \frac{h_i^{(l)} W^{(l)}}{\sqrt{|\mathcal{N}(i)|}\sqrt{|\mathcal{N}(n)|}}\right) \quad (1)$$

where $h_n^{(l)}$ is node features of n at the l^{th} layer, σ is an activation function, $\mathcal{N}(n)$ is the set of neighbors of node n, $|\mathcal{N}(n)|$ is the number of neighbors of node n, $b^{(l)}$ is the learnable bias at the l^{th} layer, and $W^{(l)}$ is the learnable weight at the l^{th} layer.

After learning is completed, node features at the final GCN layer are used for prediction. A graph pooling layer [33] is inserted at the end of the structure of Tier-predictor to create the graph representation. This representation is a two-dimensional vector, denoted as $[p_{top}, p_{bottom}]$, and it provides the probabilities of defects being in the top tier and bottom tier, respectively. Note that the proposed Tier-predictor can perform diagnosis on M3D designs with more than two tiers by extending the dimension of the graph representation vector to be the number of tiers in the CUDs. Without loss of generality, we demonstrate our framework with two-tier designs in this work. For the MIV-pinpointer, local information near the candidate MIVs is much more important than global features. Hence, node classification is used to pinpoint the set of defective MIVs. The learned node features $\{h\} \in \mathbb{R}^2$ are directly used to calculate the probability that an MIV has a defect. Note that the proposed GNN models are not restricted to M3D designs. If 2D circuits are partitioned into distinct regions, Tier-predictor can be utilized to perform region-level fault localization; MIV-pinpointer can pinpoint faulty interconnects between regions. As no change is needed for feature extraction and model construction, the proposed GNN models are expected to achieve similar results when applied to conventional 2D ICs.

TABLE III: Design matrix of M3D benchmarks. N_{sc} (N_{ch}): number of scan chains (channels); N_g: gate count; FC: fault coverage.

Design	N_g	#MIVs	N_{sc} (N_{ch})	Chain length	#Patterns	FC
AES	98K	71K	100 (5)	123	767	98.3%
Tate	187K	143K	200 (10)	171	432	98.6%
netcard	220K	173K	400 (20)	182	40438	97.3%
leon3mp	338K	250K	400 (20)	285	18737	99.1%

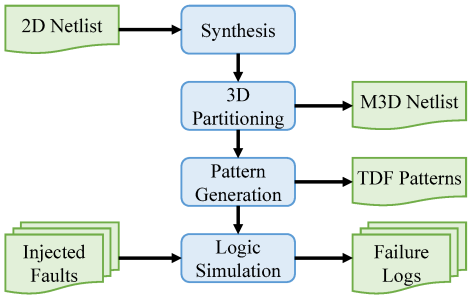


Fig. 4: Data generation flow for the proposed diagnosis framework

IV. Transferability of the Proposed GNN-Based Diagnosis Framework

In an M3D IC design flow, design configurations such as clock frequency, area, and design-for-testability (DfT) structures can result in different M3D netlists. Such differences do not change the functionality of the design; however, the way how each fault is detected might be affected during testing, leading to different failure output responses for diagnosis. Transferability on the same benchmark with different design configurations is therefore necessary for the proposed framework to be applicable to real-world scenarios. In this section, we carry out the transferability analysis on the proposed tier-level fault localization framework. We evaluate the proposed GNN-based framework on four two-tier M3D benchmarks, namely Advanced Encryption Standard (AES) and Tate Bilinear Pairing (Tate) from OpenCores, and netcard and leon3mp from the ISPD 2012 benchmark suite. We also provide a data augmentation solution to improve the transferability of our GNN models.

To evaluate the transferability of the proposed diagnosis framework, we generated datasets for each benchmark with various design configurations. Figure 4 shows our data generation flow. Given a 2D netlist at the register-transfer level (RTL), we conducted synthesis with the open-source Nangate 45 nm standard cell library using Synopsys Design Compiler. The synthesized netlists were partitioned into M3D with the partitioning algorithm proposed in [34]. Next, test-compression hardware was inserted using the embedded deterministic test (EDT) methodology (Siemens EDA Tessent), followed by the TDF pattern generation. Without loss of generality, the compaction ratio is set to 20× in all benchmarks, that is, at most 20 scan chains are connected to one test output channel using a response compactor. We also inserted bypass signals that enable the designs to scan out uncompressed responses without passing through response compactors. The design matrix of our M3D benchmarks is shown in Table III. To generate datasets for diagnosis, we randomly injected one TDF at a time in a circuit and carried out logic simulations

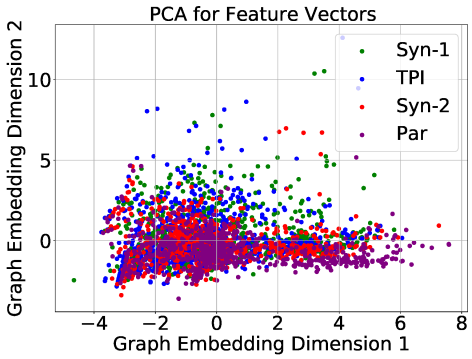


Fig. 5: Feature visualization of Tate benchmark with various design configurations.

with the TDF patterns to obtain erroneous output responses. These responses were collected into a failure log file, which created a sample in the datasets. We generated 5000 samples for each benchmark, with and without response compaction, respectively. The M3D netlist, TDF patterns, and the collection of failure logs from the data generation flow became inputs of the proposed diagnosis framework, as shown in Fig. 1.

Besides the design configuration, denoted as *Syn-1*, used for training, we generated additional datasets for each benchmark with different design configurations for the purpose of transferability analysis, including (i) *TPI*: test point (TP)-inserted netlists; (ii) *Syn-2*: netlists synthesized with another clock frequency; (iii) *Par*: netlists partitioned using the M3D partitioning algorithm in [35]. TPs are widely-used DfT structures to help in improving test coverage and reducing pattern counts. We set the maximum number of TPs to be 1% of the number of gates in the design and utilized ATPG tools to determine TP locations. Note that designs with different clock frequencies are re-synthesized and re-partitioned into M3D from the RTL level. Test patterns are re-generated after TP insertion. These changes can lead to significant variations in gate types, spatial distribution of gates, and how each fault is detected. Our goal is to create a transferable GNN-based framework that can be directly applied to these designs without retraining.

To analyze the transferability of our GNN models, we first leveraged principle component analysis (PCA) [36] to visualize the distribution of feature vectors listed in Table II. The visualization for the Tate benchmark is shown in Fig 5. Each sample represents a feature vector of a sub-graph generated by our back-tracing algorithm, where each sub-graph corresponds to an injected fault and the associated failure log. Clearly, feature distributions of all netlists, are greatly overlapped even though different design configurations lead to variations in test patterns and gate types. This is because such netlists have equivalent functionality; sub-graphs obtained from our back-tracing algorithm tend to have similar feature distributions and topological dependencies. This similarity demonstrates that the selected features and the proposed back-tracing algorithm is not biased by different design configurations when performing diagnosis on the same benchmark design. Therefore, Tier-predictor and MIV-pinpointer are able to learn from a small number of netlists and inference on others without retraining.

We further demonstrate the transferability of the proposed framework by comparing a transferred model to models trained from each design configuration. Note that the amount and the diversity of the dataset are important factors to affect the performance of ML models. To enhance the transferability of the proposed framework, we develop a data augmentation method by collecting samples from randomly-partitioned M3D netlists. As shown in Fig. 5, varying design configurations does not generate anomalous samples that can lead to an adverse impact during training. Therefore, for the purpose of training, we randomly partition 2D netlists into M3D to create various spatial distributions of logic gates. This helps in enhancing the diversity of the dataset and preventing the GNN model from being biased toward any design configuration.

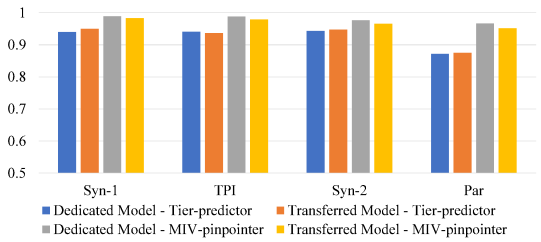


Fig. 6: Accuracy of the proposed GNN models with Tate benchmark.

Figure 6 shows the accuracy of Tier-predictor and MIV-pinpointer for the Tate benchmark, respectively. Note that *Dedicated Model* is trained individually for each design configuration, while *Transferred Model* is trained with samples from Syn-1 and two randomly-partitioned netlists. For both Tier-predictor and MIV-pinpointer, Transferred Model can achieve nearly the same accuracy as Dedicated Model. Transferred Model even outperforms Dedicated Model for the Syn-2 and Par netlists, whose samples were not included during the training of Transferred Model. This is because Transferred Model has learned from a highly-diverse training dataset obtained from randomly-partitioned netlists. Therefore, with the proposed data-augmentation solution, Tier-predictor and MIV-pinpointer can be transferred to perform diagnosis on the same benchmark with various design configurations. Such transferability is extremely important for emerging M3D technologies because the design flow is not standardized and there is a lack of post-silicon data. Reusing pre-trained models on new netlists significantly reduces the runtime for diagnosis and provides quick feedback to the foundry and design groups. These advantages help us to improve yield learning and shorten the time-to-market.

V. Proposed Candidate Pruning and Reordering Policy

In this section, we describe the proposed GNN-based candidate pruning and reordering policy. We utilize prediction results from Tier-predictor and MIV-pinpointer to enhance the quality of diagnosis reports generated by ATPG tools. Given a failure log file from the tester, the tier-level localization and ATPG diagnosis are conducted in parallel to generate prediction results and a diagnosis report, respectively. Such a diagnosis report provides a list of candidates that are predicted to be the defect locations by ATPG tools. The proposed candidate pruning and reordering policy aims to improve the

TABLE IV: Details of the confusion matrix for the proposed Tier-predictor.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (Correct prediction and Probability $\geq$ Classification threshold)	False Negative (Correct prediction and Probability $<$ Classification threshold)
Actual Negative	False Positive (Incorrect prediction and Probability $\geq$ Classification threshold)	True Negative (Incorrect prediction and Probability $<$ Classification threshold)

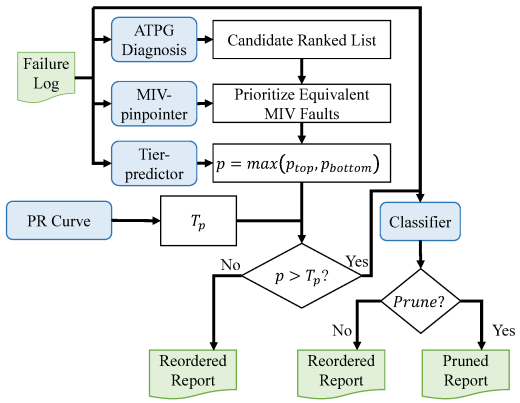


Fig. 7: Flowchart for candidate pruning and reordering.

quality of the diagnosis report by removing unlikely candidates and moving the ground-truth defect location to the top of the list.

A. Overview

An overview of the proposed candidate pruning and reordering policy is shown in Fig. 7. Given a failure log file from the tester, we utilize ATPG and our fault localization framework to produce a diagnosis report and the predicted tier-level defect locations. Next, as MIVs are prone to delay defects in M3D designs (see Section I), we first evaluate the prediction from MIV-pinpointer to extract all faulty MIVs. All candidates in the report that are equivalent to such MIVs are moved to the top of the report. This can help diagnosis engineers to prioritize MIV faults during the subsequent PFA. After evaluating MIV-pinpointer, we extract the predicted faulty tier from Tier-predictor and use its probability as a confidence score. We compare such a score to a threshold value, denoted as T_p, to determine the confidence level of the predicted faulty tier, where T_p is derived from a PR curve generated during training. If the prediction from Tier-predictor has low confidence, we reorder the diagnosis reports by moving all candidates in the faulty tier to the top of the lists. Otherwise, we utilize the proposed GNN-based *Classifier* to decide whether to prune candidates. The pruning process removes all candidates in the tier predicted to be fault-free from diagnosis reports to improve both diagnostic resolution and FHI.

B. Precision-recall Curve

In the field of classification problems, a confusion matrix is a popular visualization method to show the effectiveness of an algorithm. The description of a confusion matrix for the proposed Tier-predictor is shown in Table IV. We classify a sample as Actual Positive if the tier predicted to be faulty is equivalent to the tier-level ground truth defect location. Otherwise, the sample is classified as Actual Negative. Predicted Positive and Predicted Negative are distinguished by comparing the probability of the predicted faulty tier to a classification threshold between 0 and 1. Samples with probabilities larger than the threshold are categorized as Predicted Positive, while others are represented as Predicted Negative. By switching the threshold value, various distributions of Predictive Positive and Predictive Negative can be generated.

Based on distributions in confusion matrices, receiver operating characteristic (ROC) curves and PR curves are widely used to visualize the performance of an algorithm at all classification thresholds. Compared to ROC curves, PR curves can provide more insights when the datasets are highly imbalanced [37]. For the proposed Tier-predictor, a skew distribution between Actual Positive and Actual Negative tends to be formed as Tier-predictor can achieve up to 90% accuracy. Therefore, we choose the PR curve to evaluate our tier-level localization framework. The precision and recall are defined as follows:

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2}$$

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{3}$$

To plot PR curves, we alter the classification threshold to calculate the corresponding Precision and Recall values. As the threshold increases, Precision tends to increase while Recall tends to decrease. This is because a large threshold increases the difficulty of classifying samples as Predicted Positive, leading to a low False Positive and a high False Negative. In the proposed candidate pruning and reordering policy, samples classified as Predicted Positive (i.e., Ture Positive + False Positive) become inputs of the GNN-based Classifier. Candidates in such samples can be pruned to help in improving the quality of the corresponding diagnosis reports. However, pruning False Positive samples leads to a loss of diagnosis accuracy because the ground-truth defect locations are removed from the diagnosis reports. To avoid a large number of False Positive samples, we utilize PR curves to find the best threshold as T_P in the proposed candidate pruning and reordering policy. As our objective is to improve the quality of diagnosis reports with a loss of diagnosis accuracy below 1%, the threshold T_P is determined by calculating the minimum classification threshold in the PR curve of the training dataset such that Precision is larger than or equal to 99%.

C. GNN-based Classifier

After obtaining T_P from PR curves, samples with high-confidence Tier-predictor predictions (i.e., $\max(p_{top}, p_{bottom}) \geq T_P$) are represented as Predicted Positive. Next, we train the proposed GNN-based Classifier focusing on Predicted Positive samples to determine whether

to prune or reorder candidates in the corresponding diagnosis reports. Note that the pruning process can improve both diagnostic resolution and FHI for True Positive samples without any loss of diagnosis accuracy. For False Positive samples, the ground-truth defect locations are removed due to incorrect Tier-predictor predictions, leading to an adverse impact on the subsequent PFA. The objective of the proposed Classifier is to prioritize Predicted Positive samples and to distinguish True Positive from False Positive. This allows the pruning process to significantly improve the quality of diagnosis reports while minimizing accuracy loss. In addition, T_P obtained during training may not be the best classification threshold for diagnosing netlists with various design configurations, leading to an increase in the number of False Positive samples. The transferability of Classifier helps in extracting such samples appropriately to prevent any significant loss in diagnosis accuracy.

For the proposed Classifier, an imbalanced dataset is a major challenge during training. Because our Tier-predictor can achieve extremely high accuracy, the number of True Positive samples is significantly larger than False Positive samples. From our experiments, a ratio of around 90:1 is observed for the Tate benchmark. Such an unbalanced dataset leads to a distorted model that tends to ignore the minority class. Oversampling is a common technique used to increase the size of the minority class and make it similar to the size of the majority class. Several oversampling algorithms have been proposed in previous work [38] [39]. However, existing algorithms cannot be directly applied to graphs because they mainly focus on Euclidean data. A conversion step is required to map graph data to simplified representations, while such a conversion can lead to a loss of topological dependencies in the graphs. Therefore, we develop a novel oversampling algorithm by inserting dummy buffers into samples in the minority class. For each sample, we append one buffer at the output of each node once at a time to create synthetic samples. The synthetic samples help to increase the population of the minority class without affecting the functionality of CUDs. During the oversampling process, consecutive buffers are added to each node to generate minority data until the dataset becomes balanced.

With the balanced dataset, we leverage the network-based deep transfer learning [40] to train the proposed Classifier. We first append the pre-trained hidden layers from Tier-predictor to Classifier, followed by trainable classification layers. Pre-trained layers are used to extract informative features from Tier-predictor; classification layers are responsible for determining whether to prune or reorder the reports according to the extracted features. A graph pooling layer is inserted at the end of Classifier to generate the probabilities of pruning and reordering. Such probabilities guide in how to fine-tune the diagnosis reports to improve the quality of the diagnosis process.

D. ATPG Report Pruning and Reordering

Using the results from our GNN models, we prune and reorder candidates in the ATPG diagnosis report to improve the diagnostic resolution and the FHI. Fig. 8 presents an example of our pruning and reordering process. We first collect all candidates listed in the diagnosis report generated by ATPG. Results of the MIV-pinpointer are then analyzed to extract candidate fault sites in the diagnosis report that are equivalent to the MIVs predicted to be faulty. Such fault sites are placed at the top of the final report to prioritize MIV faults during the subsequent failure analysis. As MIVs are prone to defects in emerging M3D integration [9], FHI can be improved in this way. Next, the maximum of $[p_{top}, p_{bottom}]$, denoted as p, is compared with T_p to determine the confidence level of the Tier-predictor prediction. If such a prediction has high confidence (i.e., $p \geq T_P$), we utilize Classifier to decide whether to prune or reorder candidates. Otherwise, the reordered report is generated.

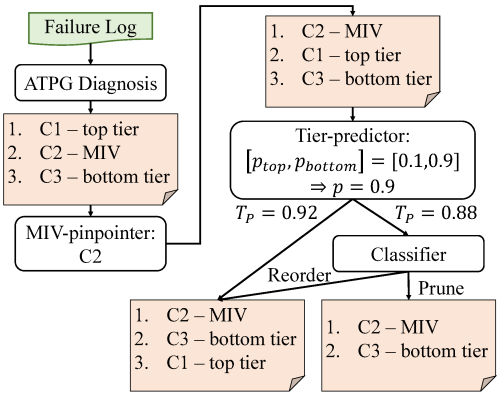


Fig. 8: Example of the ATPG report pruning and reordering process.

To reorder candidates, all fault sites in the top (bottom) tier are moved toward the top of the report if $p_{top} > p_{bottom}$ ($p_{bottom} > p_{top}$). For pruning, candidates in the tier predicted to be fault-free are filtered out from the final report because such candidates are unlikely to be the ground-truth fault location. Note that filtering out candidates may occasionally lead to a loss of accuracy. However, when Tier-predictor points out the incorrect tier as being faulty, the accuracy loss may be recovered by the MIV-pinpointer if the ground-truth fault is equivalent to the MIV fault localized by the MIV-pinpointer. With the pruning and reordering process, the ground-truth defect locations can be placed near the top of the lists, leading to better FHI than the ATPG diagnosis reports. Pruning also helps in improving diagnostic resolution by reducing the number of redundant candidates.

VI. Experimental Results

A. Diagnosis on Benchmark M3D designs

We first examine the quality of our diagnosis framework using 750 samples (15% of the generated dataset) for benchmark M3D designs with various design configurations. The quality of ATPG diagnosis reports is shown in Table V. We compare diagnostic resolution, accuracy, and FHI of diagnosis reports obtained after the pruning and reordering step with a baseline fault localization algorithm in [11], which has been demonstrated to achieve significant improvement in diagnostic resolution for conventional 2D designs. Note that in the experimental results section in [11], only the results from

TABLE V: Quality of ATPG diagnosis reports for M3D benchmarks without response compaction.

Design	Configuration	Accuracy	Mean diagnostic resolution	Standard deviation diagnostic resolution	Mean FHI	Standard deviation FHI
AES	Syn-1	100.0%	5.2	5.5	4.1	4.2
	TPI	100.0%	5.3	5.6	4.0	4.5
	Syn-2	100.0%	6.0	6.2	4.0	4.2
	Par	100.0%	5.7	5.7	3.7	3.9
Tate	Syn-1	100.0%	4.4	4.6	3.6	4.0
	TPI	100.0%	4.6	5.0	3.7	4.2
	Syn-2	100.0%	4.6	5.3	3.7	4.5
	Par	100.0%	4.1	5.4	2.8	2.6
netcard	Syn-1	96.4%	28.1	28.4	19.0	21.8
	TPI	100.0%	10.3	8.6	7.0	6.6
	Syn-2	97.4%	31.9	28.5	21.5	23.4
	Par	88.6%	21.0	21.9	12.7	17.0
leon3	Syn-1	98.8%	14.0	17.8	9.8	13.8
	TPI	99.8%	10.8	11.6	7.3	8.5
	Syn-2	98.8%	12.5	16.0	8.0	10.3
	Par	97.8%	11.5	17.2	8.5	15.5

TABLE VI: Effectiveness of delay fault-localization in M3D benchmarks without response compaction using a 2D baseline [11] and the proposed framework

Config.	[11]				Proposed framework						
					GNN standalone			GNN + [11]			
	Acc.	μ (σ) resol.	μ (σ) FHI	Tier local.	Acc.	μ (σ) resol.	μ (σ) FHI	Acc.	μ (σ) resol.	μ (σ) FHI	Tier local.
AES											
Syn-1	100.0% (-0.0%)	2.5 (1.7) (+51.9%)	2.1 (1.5) (+48.8%)	67.9%	99.2% (-0.8%)	4.9 (5.5) (+5.8%)	3.3 (3.3) (+19.5%)	99.2% (-0.8%)	2.5 (1.8) (+51.9%)	2.1 (1.5) (+48.8%)	85.5%
TPI	100.0% (-0.0%)	2.7 (1.9) (+49.1%)	2.1 (1.7) (+47.5%)	62.5%	99.7% (-0.3%)	5.1 (5.7) (+3.8%)	3.1 (3.5) (+22.5%)	99.7% (-0.3%)	2.6 (1.9) (+50.9%)	2.1 (1.8) (+47.5%)	87.4%
Syn-2	100.0% (-0.0%)	3.4 (3.1) (+43.3%)	2.5 (2.5) (+37.5%)	56.9%	99.6% (-0.4%)	5.9 (6.2) (+1.7%)	3.2 (3.3) (+20.0%)	99.6% (-0.4%)	3.4 (3.2) (+43.3%)	2.5 (2.5) (+37.5%)	86.6%
Par	99.9% (-0.1%)	3.5 (3.2) (+38.6%)	2.5 (2.4) (+32.4%)	49.4%	100.0% (0.0%)	5.6 (5.7) (+1.8%)	3.2 (3.5) (+13.5%)	99.9% (-0.1%)	3.5 (3.2) (+38.6%)	2.5 (2.4) (+32.4%)	73.7%
Tate											
Syn-1	99.9% (-0.1%)	3.0 (2.3) (+31.8%)	2.3 (1.8) (+36.1%)	42.6%	99.9% (-0.1%)	4.1 (4.2) (+6.8%)	2.7 (2.9) (+25.0%)	99.7% (-0.3%)	2.9 (2.3) (+34.1%)	2.2 (1.7) (+38.9%)	91.2%
TPI	100.0% (-0.0%)	2.9 (2.5) (+37.0%)	2.3 (1.9) (+37.8%)	41.1%	99.3% (-0.7%)	4.2 (4.6) (+8.7%)	2.6 (2.6) (+29.7%)	99.3% (-0.7%)	2.8 (2.5) (+39.1%)	2.2 (1.9) (+40.5%)	91.5%
Syn-2	100.0% (-0.0%)	3.2 (3.1) (+30.4%)	2.5 (2.5) (+32.4%)	37.2%	99.6% (-0.4%)	4.2 (5.0) (+8.7%)	2.7 (2.9) (+27.0%)	99.6% (-0.4%)	3.1 (3.0) (+32.6%)	2.4 (2.4) (+35.1%)	91.5%
Par	100.0% (-0.0%)	3.0 (2.8) (+26.8%)	2.1 (1.8) (+25.0%)	37.2%	99.0% (-1.0%)	4.0 (5.4) (+2.4%)	2.4 (2.7) (+14.3%)	99.0% (-1.0%)	2.9 (2.8) (+29.3%)	2.1 (1.8) (+25.0%)	82.8%
netcard											
Syn-1	96.4% (-0.0%)	26.2 (27.6) (+6.8%)	17.8 (20.9) (+6.3%)	1.5%	96.0% (-0.4%)	18.9 (20.2) (+32.7%)	10.9 (14.0) (+42.6%)	96.0% (-0.4%)	18.0 (19.7) (+35.9%)	10.6 (13.8) (+44.2%)	95.4%
TPI	100.0% (-0.0%)	7.6 (6.5) (+26.2%)	5.3 (5.3) (+24.3%)	6.1%	99.8% (-0.2%)	7.5 (7.5) (+27.2%)	4.2 (5.1) (+40.0%)	99.5% (-0.5%)	4.6 (5.7) (+55.3%)	3.0 (3.9) (+57.1%)	98.1%
Syn-2	97.4% (-0.0%)	22.7 (23.4) (+28.8%)	15.1 (18.1) (+29.8%)	7.8%	97.2% (-0.2%)	22.9 (21.7) (+28.2%)	12.3 (14.2) (+42.8%)	97.2% (-0.2%)	20.0 (19.6) (+37.3%)	10.7 (12.8) (+50.2%)	95.9%
Par	88.6% (-0.0%)	18.8 (20.7) (+10.5%)	12.2 (16.9) (+3.9%)	2.3%	87.1% (-1.5%)	15.2 (16.6) (+27.6%)	7.7 (10.1) (+39.4%)	87.1% (-1.5%)	14.3 (15.9) (+31.9%)	7.3 (9.8) (+42.5%)	94.5%
leon3mp											
Syn-1	98.8% (-0.0%)	11.2 (16.4) (+20.0%)	7.7 (12.5) (+21.4%)	5.3%	97.8% (-1.0%)	9.7 (12.5) (+30.7%)	6.2 (9.2) (+36.7%)	97.8% (-1.0%)	8.8 (12.0) (+37.1%)	5.7 (8.8) (+41.8%)	93.5%
TPI	99.8% (-0.0%)	8.2 (10.0) (+24.1%)	5.5 (7.5) (+24.7%)	2.9%	99.0% (-0.8%)	8.2 (9.8) (+24.1%)	4.9 (6.8) (+32.9%)	98.8% (-1.0%)	6.9 (9.4) (+36.1%)	4.3 (6.3) (+41.1%)	88.2%
Syn-2	98.8% (-0.0%)	9.6 (14.3) (+23.2%)	6.0 (9.0) (+25.0%)	2.7%	98.1% (-0.7%)	8.8 (11.1) (+29.6%)	5.1 (6.8) (+36.3%)	98.1% (-0.7%)	8.0 (10.6) (+36.0%)	4.6 (6.3) (+42.5%)	91.8%
Par	97.8% (-0.0%)	9.9 (16.3) (+13.9%)	7.4 (14.6) (+12.9%)	8.7%	94.9% (-2.9%)	8.2 (11.9) (+28.7%)	5.2 (8.6) (+38.8%)	94.9% (-2.9%)	7.8 (11.7) (+32.2%)	5.0 (8.4) (+41.2%)	86.8%

Config.: configuration; Acc.: accuracy; μ: mean; σ: standard deviation; resol.: diagnostic resolution; local.: localization.

the first-level classifier in the proposed two-level classification framework are chosen to prevent a large loss of accuracy. As diagnosis accuracy is also the top priority during the candidate pruning and reordering process, we compare the results from the first-level classifier in the baseline with our framework.

Table VI shows the results for each benchmark without response compaction, where the values in parenthesis are changes from the ATPG diagnosis reports listed in Table V. Note that the GNN models in our experiments are trained with datasets from Syn-1 and two randomly-partitioned netlists. Netlists with different configurations are evaluated to demonstrate the transferability of our framework. For AES and Tate,

TABLE VII: Quality of ATPG diagnosis reports for M3D benchmarks with response compaction.

Design	Configuration	Accuracy	Mean diagnostic resolution	Standard deviation diagnostic resolution	Mean FHI	Standard deviation FHI
AES	Syn-1	98.8%	9.0	18.2	5.3	10.5
	TPI	96.9%	10.6	20.7	4.8	8.0
	Syn-2	97.1%	12.9	21.9	5.6	10.4
	Par	98.4%	11.7	20.0	5.7	10.9
Tate	Syn-1	99.9%	5.7	10.0	4.1	5.5
	TPI	100.0%	5.8	10.4	4.2	5.9
	Syn-2	99.4%	6.7	14.0	4.5	7.2
	Par	99.7%	6.1	14.4	3.4	6.1
netcard	Syn-1	94.8%	30.6	30.6	20.3	22.9
	TPI	99.7%	11.8	12.8	8.0	8.9
	Syn-2	93.5%	34.3	30.8	21.0	22.5
	Par	87.8%	23.6	25.4	14.0	18.4
leon3	Syn-1	98.5%	14.8	19.1	10.0	13.8
	TPI	99.8%	11.8	14.7	8.1	11.2
	Syn-2	98.8%	13.1	16.7	8.4	10.8
	Par	97.5%	12.0	18.0	8.6	15.3

TABLE VIII: Effectiveness of delay fault-localization in M3D benchmarks with response compaction using a 2D baseline [11] and the proposed framework.

Config.	[11]				Proposed framework						
					GNN standalone			GNN + [11]			
	Acc.	μ (σ) resol.	μ (σ) FHI	Tier local.	Acc.	μ (σ) resol.	μ (σ) FHI	Acc.	μ (σ) resol.	μ (σ) FHI	Tier local.
AES											
Syn-1	98.8% (-0.0%)	4.9 (9.4) (+45.6%)	2.6 (4.9) (+50.9%)	47.4%	98.4% (-0.4%)	8.2 (17.5) (+8.9%)	4.0 (8.5) (+24.5%)	98.4% (-0.4%)	4.6 (9.0) (+48.9%)	2.5 (4.7) (+52.8%)	85.7%
TPI	96.9% (-0.0%)	6.0 (11.5) (+43.4%)	2.4 (3.8) (+50.0%)	45.6%	96.3% (-0.6%)	10.0 (20.5) (+5.7%)	4.1 (8.0) (+14.6%)	96.3% (-0.6%)	5.8 (11.6) (+45.3%)	2.4 (3.8) (+50.0%)	82.0%
Syn-2	97.1% (-0.0%)	7.9 (12.8) (+38.8%)	3.2 (6.4) (+42.9%)	32.1%	95.4% (-1.7%)	12.2 (21.6) (+5.4%)	5.1 (10.0) (+8.9%)	95.4% (-1.7%)	7.5 (12.5) (+41.9%)	3.2 (6.6) (+42.9%)	75.6%
Par	98.4% (-0.0%)	7.5 (11.3) (+35.9%)	3.3 (5.5) (+42.1%)	33.7%	94.8% (-3.6%)	10.7 (19.1) (+8.5%)	4.6 (8.4) (+19.3%)	94.8% (-3.6%)	6.9 (10.9) (+41.0%)	3.1 (4.9) (+45.6%)	74.8%
Tate											
Syn-1	99.9% (-0.0%)	3.7 (5.5) (+35.1%)	2.6 (3.3) (+36.6%)	38.6%	99.2% (-0.7%)	5.4 (9.9) (+5.3%)	3.2 (5.1) (+22.0%)	99.2% (-0.7%)	3.6 (5.5) (+36.8%)	2.5 (3.2) (+39.0%)	83.6%
TPI	100.0% (-0.0%)	3.6 (5.8) (+37.9%)	2.5 (2.9) (+40.5%)	38.3%	99.6% (-0.4%)	5.4 (10.3) (+6.9%)	3.3 (5.3) (+21.4%)	99.6% (-0.4%)	3.5 (5.8) (+39.7%)	2.4 (2.9) (+42.9%)	84.3%
Syn-2	99.4% (-0.0%)	4.5 (8.7) (+32.8%)	2.9 (4.5) (+35.6%)	30.7%	99.2% (-0.2%)	6.4 (14.0) (+4.5%)	3.7 (7.6) (+17.8%)	99.2% (-0.2%)	4.4 (8.8) (+34.3%)	2.8 (4.5) (+37.8%)	80.3%
Par	99.7% (-0.0%)	4.1 (7.5) (+32.8%)	2.5 (4.0) (+26.5%)	33.3%	98.7% (-1.0%)	5.9 (14.4) (+3.3%)	3.5 (8.2) (+2.9%)	98.7% (-1.0%)	4.0 (7.6) (+34.4%)	2.5 (4.0) (+26.5%)	73.8%
netcard											
Syn-1	94.8% (-0.0%)	29.5 (30.1) (+3.6%)	19.4 (22.3) (+4.4%)	0.2%	94.4% (-0.4%)	22.9 (26.8) (+25.2%)	12.3 (16.3) (+39.4%)	94.4% (-0.4%)	22.5 (26.6) (+26.5%)	12.1 (16.2) (+40.4%)	93.2%
TPI	99.7% (-0.0%)	5.6 (9.4) (+52.5%)	4.3 (6.6) (+46.2%)	32.7%	98.7% (-1.0%)	8.2 (10.6) (+30.5%)	4.7 (6.5) (+41.2%)	98.7% (-1.0%)	5.2 (8.3) (+55.9%)	3.5 (5.9) (+56.2%)	96.0%
Syn-2	93.5% (-0.0%)	29.8 (28.6) (+13.1%)	17.7 (19.8) (+15.7%)	1.4%	92.2% (-1.3%)	24.6 (26.4) (+28.3%)	12.1 (14.3) (+42.4%)	92.2% (-1.3%)	22.9 (25.5) (+33.2%)	11.2 (13.7) (+46.7%)	95.0%
Par	87.8% (-0.0%)	20.0 (23.8) (+15.3%)	12.8 (17.8) (+8.6%)	5.1%	86.3% (-1.5%)	20.2 (24.0) (+14.4%)	9.6 (13.2) (+31.4%)	86.3% (-1.5%)	18.6 (23.2) (+21.2%)	9.0 (12.7) (+35.7%)	86.5%
leon3mp											
Syn-1	98.6% (-0.1%)	10.7 (17.1) (+27.7%)	7.1 (12.1) (+29.0%)	13.1%	97.0% (-1.5%)	10.6 (14.9) (+28.4%)	6.3 (9.2) (+37.0%)	97.0% (-1.5%)	9.4 (14.2) (+36.5%)	5.6 (8.7) (+44.0%)	91.2%
TPI	99.8% (-0.0%)	6.3 (11.8) (+46.6%)	4.5 (9.2) (+44.4%)	29.9%	97.8% (-2.0%)	9.3 (13.4) (+21.2%)	5.3 (8.3) (+34.6%)	97.8% (-2.0%)	8.1 (13.2) (+31.4%)	5.1 (9.6) (+37.0%)	86.8%
Syn-2	98.8% (-0.0%)	10.2 (14.9) (+22.1%)	6.4 (9.4) (+23.8%)	2.4%	97.6% (-1.2%)	9.2 (12.0) (+29.8%)	5.3 (7.3) (+36.9%)	97.6% (-1.2%)	8.4 (11.7) (+35.9%)	4.9 (6.9) (+41.7%)	91.0%
Par	97.5% (-0.0%)	8.4 (15.8) (+30.0%)	6.4 (13.1) (+25.6%)	24.3%	93.8% (-3.7%)	10.3 (16.9) (+14.2%)	6.2 (11.6) (+27.9%)	93.8% (-3.7%)	9.5 (16.5) (+20.8%)	5.9 (11.2) (+31.4%)	76.3%

the improvement in diagnostic resolution and FHI of the proposed framework with GNN standalone are less obvious than the improvement obtained from the baseline. This is because ATPG diagnosis reports tend to provide good resolution with candidates only in the faulty tier. Therefore, the candidate pruning and reordering process does not considerably benefit from the tier-level localization to improve the quality of diagnosis reports. In contrast, the baseline approach analyzes each candidate one at a time to determine whether such a candidate should be removed from the diagnosis report. It is expected that the baseline can achieve better improvement in diagnostic resolution and FHI than the proposed framework for

small benchmarks. However, for netcard and leon3mp, ATPG reports are likely to contain candidates in both tiers as the complexity of designs increases. By pruning and reordering candidates based on the predictions of our GNN models, the proposed framework can enhance the quality of diagnosis reports more significantly than the baseline approach, without any unacceptable loss in accuracy.

An important advantage of the proposed framework is its compatibility with existing algorithms for conventional 2D designs to provide the resolution at the tier level and further improve the quality of diagnosis reports. We first utilize the proposed framework to carry out tier-level localization, followed by the baseline approach to evaluate the remaining candidates after the pruning and reordering process. Compared to the results with standalone GNN and the baseline, the combined approach improves the diagnostic resolution and FHI without additional loss of accuracy. Such improvements can achieve more than 55% for the netcard benchmark with the TPI configuration. As the proposed framework can be carried out alongside the ATPG diagnosis and the candidate pruning and reordering process can be embedded in the candidate analysis step, no test time overhead is needed to combine the proposed framework with the baseline approach.

Note that the proposed candidate pruning and reordering policy may occasionally remove the ground-truth defect location from the diagnosis report. To compensate for the loss of accuracy, we generate a backup dictionary, which records the candidates being pruned corresponding to each failure chip. Diagnosis engineers can therefore search in the backup dictionary for further analysis whenever the root cause of a failure is not found based on the pruned report. With this compensation method, our framework is guaranteed to achieve the same accuracy as ATPG. Although the backup dictionary requires additional memory, its size depends on the number of candidates being pruned in each sample, which can be estimated by the difference in diagnostic resolution between ATPG diagnosis reports and reports generated by the proposed framework. As shown in Table VI, the largest difference among the four benchmarks is Syn-2 for netcard; the size of the corresponding backup dictionary is only 246 kilobytes. Therefore, the memory overhead of the proposed method is within acceptable limits.

In addition to diagnostic resolution and FHI, the resolution at the tier level during diagnosis is important for M3D designs to facilitate yield learning. As the baseline approach does not directly provide such a resolution, tier-level localization can be achieved by analyzing the remaining candidates after fault localization. If all candidates are in the faulty tier, the corresponding report is successfully localized at the tier level; otherwise, tier-level localization is not accomplished. For the proposed framework, tier-level localization is obtained from the predictions of our Tier-predictor. Tier localization values in Table VI are the percentages of reports being localized at the faulty tier using the baseline and the proposed framework, respectively. Note that we do not consider the reports that have been localized during the ATPG diagnosis process (i.e., ATPG diagnosis reports only contain candidates in one tier) in the calculation. Clearly, the proposed framework can accurately identify the faulty tier for all benchmark M3D designs with various design configurations. The baseline approach is not effective for tier-level localization because the tier structure and fabrication-related defects in M3D do not exist in 2D designs. Therefore, the tier-level location of each candidate is overlooked during fault localization. The effectiveness of the baseline approach on tier-level localization decreases when the size of the CUD becomes large as the number of candidates increases in diagnosis reports. Although the diagnostic resolution and FHI are significantly improved, candidates after fault localization are likely to be located in both tiers. The evaluation of tier localization demonstrates that the baseline approach is not sufficient to provide a high level of resolution at the tier level for M3D designs. With the proposed framework, the faulty tier can be identified early in the diagnosis process to provide quick feedback to the foundry and facilitate yield learning.

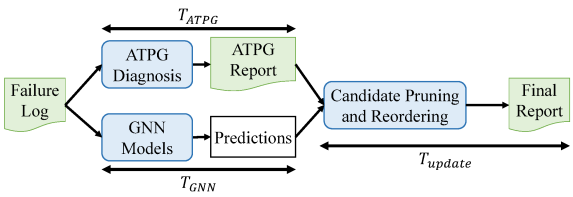


Fig. 9: Runtime for the deployment of the proposed framework.

The quality of ATPG diagnosis reports and results for benchmark designs with response compaction are shown in Table VII and Table VIII, respectively. Note that both the diagnostic resolution and accuracy of designs are worse than the results without compaction. This is expected because the scan cells that capture erroneous responses cannot be pinpointed without bypass signals. The search space is therefore increased, leading to a reduction in diagnostic resolution and accuracy. However, the proposed framework is shown to be effective with compressed patterns. Reports generated by this framework achieve up to 30.5% improvement in diagnostic resolution and 42.4% improvement in FHI with a very low accuracy loss. For tier-level localization, the proposed framework localizes faults at the tier level more effectively than the baseline approach. Furthermore, our approach does not require additional hardware or test data and is compatible with any combinational (e.g., XOR-based) response compactor. Evaluation of netlists with different configurations clearly demonstrates the transferability of our framework. This advantage is significant for the emerging M3D technology as there is no standardized design flow. Therefore, M3D netlists with various synthesis and partitioning results can be generated. GNN models in our framework can be transferable to perform diagnosis directly on such netlists without retraining.

B. Runtime Analysis

We next conduct the runtime analysis of the proposed framework, including the training phase and the framework deployment. The training phase contains pre-processing steps for feature construction and the training processes of the proposed GNN models. The runtime for the deployment of our framework is shown in Fig. 9. Given a failure log file, ATPG diagnosis and GNN model inferencing are carried out

TABLE IX: Runtime analysis of the proposed framework for benchmark M3D designs.

Design	Training		Deployment		
	Feature construction (sec)	GNN training (sec)	T_{ATPG} (sec)	T_{GNN} (sec)	T_{update} (sec)
AES	1.8k	1.8k	152.7	55.4	11.0
Tate	2.0k	906.9	340.6	51.1	11.6
netcard	213.9k	823.3	536.8	227.7	15.4
leon3mp	140.9k	1.7k	1.0k	252.7	3.8

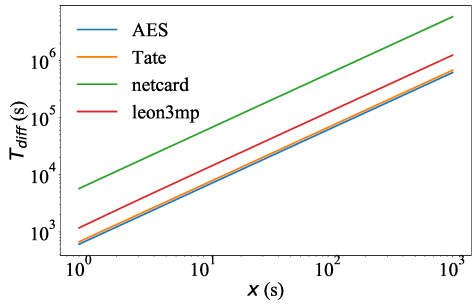


Fig. 10: Difference in runtime for PFA between the ATPG flow and the proposed framework.

simultaneously, followed by the candidate pruning and reordering process to update the ATPG diagnosis report. Table IX shows the runtime for training and deploying the proposed framework. Note that the values for the deployment are the total runtime needed to complete diagnosing test sets (i.e., 750 samples) for benchmarks with Syn-2 design configuration. In the training phase, the runtime for the feature construction is proportional to the size and the number of test patterns of the CUD. However, feature construction is required to be conducted only once and can be reused for every failure log file during inferencing; therefore, the cost can be amortized. For the framework deployment, GNN model inferencing is much faster than the ATPG diagnosis process. The runtime overhead of adding our framework on commercial tools is the subsequent candidate pruning and reordering process, which is relatively small compared to T_{ATPG}.

To quantify the effectiveness of the proposed framework on the subsequent PFA, we assume that the PFA requires x seconds to analyze one candidate in a diagnosis report. Let T_{total} be the total runtime to identify the ground-truth defect location. For the ATPG flow, T_{total}(ATPG) can be calculated as $T_{ATPG} + FHI_{ATPG} \times x$, where FHI_{ATPG} is the FHI in the ATPG diagnosis report. While for the proposed framework, T_{total}(proposed framework) equals to $\max(T_{ATPG}, T_{GNN}) + T_{update} + FHI_{update} \times x$, where FHI_{update} is the updated FHI in the final report after candidate pruning and reordering. The difference $T_{diff} = T_{total}(\text{ATPG}) - T_{total}(\text{proposed framework})$ over the test set for each benchmark with Syn-2 configuration is shown in Fig. 10. Note that positive T_{diff} demonstrates that leveraging the proposed framework can save time for identifying the ground truth defects compared to the ATPG tool. As the runtime for each candidate in the PFA process increases, the amount of time saved from the proposed framework becomes more significant. Our framework is most effective in reducing runtime for the netcard benchmark because the improvement in FHI after candidate pruning and reordering can achieve up to 42.7%. For other benchmarks, T_{diff} is at least 10^3 seconds when x is larger than 10 seconds. The reduction in runtime for PFA enables us to efficiently identify the root causes of defective chips. This is important for the emerging M3D integration technology to increase the manufacturing processes and shorten the time-to-market.

VII. Discussion

A. Diagnosis on Designs with Multiple Faults

In M3D, different tiers suffer from different fabrication-related systematic defects due to immature manufacturing processes (see Section I). Such defects tend to cause multiple delay faults throughout the faulty tier, which significantly impacts the timing of the circuit and increases the complexity of logic diagnosis. Tier-level localization becomes important to identify the faulty tier early in the diagnosis process to accelerate yield learning. Therefore, the proposed framework is extended to perform diagnosis on defective designs that have multiple faults in the faulty tier.

To simulate the designs with tier-specific systematic defects, we randomly inject 2 to 5 TDFs in one tier and carry out fault simulation to create the failure log file. We generate 5000 failure log files for each benchmark with Syn-1 configuration for the purpose of training. The testing datasets are composed of 750 samples for benchmarks with Syn-2 configuration to demonstrate the effectiveness and transferability of the proposed framework. Results of multiple delay faults localization are shown in Table X. Note that a diagnosis report is counted as accurate if all injected faults in the CUD are included in the candidate list. ATPG reports for the netcard benchmark suffer from low accuracy because the number of test patterns for netcard is much more than other benchmarks. Injecting multiple faults in the design leads to failure log files with a large number of failing patterns and failure output responses, increasing the searching space and making it difficult to accurately narrow down candidate fault locations. Although diagnosis accuracy is limited by ATPG reports, the proposed framework provides the resolution at the tier level with 88.0% accuracy. This advantage compensates for the low diagnosis accuracy as the foundry can review its manufacturing processes directly based on predictions of the proposed Tier-predictor even if the ground-truth defect locations cannot be identified from the diagnosis reports. Moreover, the quality of diagnosis reports for all benchmarks is enhanced by the improvement in FHI. Such an improvement prevents the subsequent PFA from wasting time on analyzing fault-free candidates. The analysis of multiple faults localization demonstrates that the proposed framework is applicable to perform diagnosis on designs with tier-specific systematic defects. As such defects are needed to be eliminated before M3D can become ready for commercial exploitation, feedback from the proposed framework is essential for the foundry to accelerate yield learning and improve the immature fabrication processes.

B. Diagnosis with Standalone Tier-predictor and MIV-pinpointer

In the proposed framework, Tier-predictor is utilized to predict the faulty tier and provide guidance on the candidate

TABLE X: Effectiveness of multiple delay faults-localization in M3D benchmarks using ATPG diagnosis and the proposed framework.

Design	ATPG diagnosis only					Proposed framework					
	Accuracy	Mean resol.	Std. resol.	Mean FHI	Std. FHI	Accuracy	Mean resol.	Std. resol.	Mean FHI	Std. FHI	Tier local.
AES	99.7%	6.0	3.6	3.8	2.3	99.6% (-0.1%)	5.9 (+1.1%)	3.6	3.4 (+12.3%)	2.1	64.3%
Tate	97.7%	5.1	3.6	3.9	2.9	97.1% (-0.6%)	4.8 (+5.9%)	3.3	3.0 (+23.0%)	2.1	78.1%
netcard	53.2%	35.5	14.6	18.2	13.2	52.8% (-0.4%)	17.4 (+31.6%)	11.3	11.4 (+37.3%)	9.0	88.0%
leon3mp	88.1%	12.6	8.7	7.9	5.4	88.1% (-0.0%)	12.6 (+0.0%)	8.7	5.7 (+28.7%)	4.6	79.0%

TABLE XI: Effectiveness of delay fault-localization with individual models of the proposed framework.

Diagnosis method	Accuracy	Mean resol.	Std. resol.	Mean FHI	Std. FHI
ATPG only	100.0%	4.9	5.3	3.6	4.2
Tier-predictor	98.5% (-1.5%)	4.6 (+6.1%)	5.3	2.9 (+19.4%)	3.2
MIV-pinpointer	100.0% (-0.0%)	4.9 (+0.0%)	5.3	3.6 (+0.0%)	4.2
Tier-predictor + MIV-pinpointer	99.1% (-0.9%)	4.7 (+4.1%)	5.3	2.9 (+19.4%)	3.2

pruning and reordering process; MIV-pinpointer aims at identifying faulty MIVs. To evaluate their impacts on improving the quality of diagnosis reports, we carry out a detailed analysis by performing diagnosis with each model standalone. Note that in order to clearly demonstrate the influence of MIV-pinpointer, we augment the size of the test set by 10% with MIV fault-injected samples only.

Table XI shows the effectiveness of fault localization with individual models and with both models on the AES benchmark with Syn-1 configuration. Tier-predictor achieves better improvement in diagnostic resolution and FHI than MIV-pinpointer as the candidate pruning and reordering process improves the quality of reports mainly based on tier-level predictions. However, diagnosis with Tier-predictor standalone suffers from more than 1% loss of diagnosis accuracy. This is because MIVs do not belong to any tiers in the M3D designs. If an MIV fault is presented, the candidate pruning process may occasionally remove such a faulty MIV from the diagnosis report when the prediction from MIV-pinpointer is not considered. In contrast, the changes in the quality of diagnosis reports from MIV-pinpointer are not obvious because MIV-pinpointer standalone only moves the predicted faulty MIVs to the top of reports without pruning or reordering the remaining candidates. No improvement is achieved if such faulty MIVs have already been placed at the top of candidate lists during ATPG diagnosis. However, predictions from MIV-pinpointer can compensate for the loss of accuracy caused by Tier-predictor as we prioritize MIV faults when updating the diagnosis reports. If an MIV is predicted to be faulty, it can no longer be pruned by the subsequent candidate pruning and reordering process. This effect can be clearly observed by the results when both Tier-predictor and MIV-pinpointer are applied. Compared to Tier-predictor standalone, the loss of accuracy is improved to be below 1% when MIV-pinpointer helps in reserving susceptible MIV faults in the reports, which is significant to prevent misleading candidates from the PFA. Therefore, both models are of the same importance in the proposed framework to achieve tier-level localization and improve the quality of diagnosis reports.

VIII. Conclusion

We have proposed a GNN-based framework to conduct tier-level fault diagnosis simply based on the CUD netlist and failure log files from the tester. Two GNN models, namely Tier-predictor and MIV-pinpointer, have been trained to predict which tier and MIVs have defects. We have conducted the transferability analysis between various design configurations and proposed a data-augmentation method to improve the transferability of the proposed framework. We have also provided a GNN-based candidate reordering and pruning algorithm using our predictions to improve the quality of ATPG diagnosis reports. We have shown that with a very low accuracy loss, the diagnostic resolution and the FHI are significantly improved for the OpenCore and ISPD benchmarks. We have demonstrated that our framework is effective for designs with test compression without additional resource requirements, and it is compatible with commercial tools and existing algorithms to provide the high level of resolution at the tier level. We have discussed the transferability of the proposed framework between benchmark M3D designs and provided guidance on choosing appropriate models for diagnosis. We have also provided a detailed analysis of performing diagnosis with individual Tier-predictor and MIV-pinpointer and highlighted their impacts on improving the quality of diagnosis reports.

IX. Acknowledgement

The authors would like to thank C. Papameletis and S. Chillarige of Cadence Design Systems for their helpful suggestions.

References

[1] S. Panth et al. Design challenges and solutions for ultra-high-density monolithic 3D ICs. In *SOI-3D-Subthreshold Microelectronics Technology Unified Conference*, pages 1–2, 2014.

[2] P. Batude et al. 3D sequential integration: A key enabling technology for heterogeneous co-integration of new function with CMOS. *IEEE J. Emerg. Sel*, 2(4):714–722, Dec 2012.

[3] A. Mokhberi et al. A comparative study of dopant activation in boron, BF/sub 2/, arsenic, and phosphorus implanted silicon. *IEEE Trans. Electron Devices*, 49(7):1183–1191, 2002.

[4] A. Vandooren et al. Sequential 3D: Key integration challenges and opportunities for advanced semiconductor scaling. In *Interational Conference on IC Design Technology*, pages 145–148, 2018.

[5] A. Mallik et al. The impact of sequential-3D integration on semiconductor scaling roadmap. In *IEEE IEDM*, pages 32.1.1–31.1.4, 2017.

[6] M. Brocard et al. Impact of intermediate BEOL technology on standard cell performances of 3D VLSI. In *European Solid-State Device Research Conference*, pages 218–221, 2016.

[7] C.Fenouillet Beranger et al. W and Copper interconnection stability for 3D VLSI CoolCube integration. In *Inter. conf. Solid State Devices and Materials*, 2015.

[8] K. Garidis et al. Characterization of bonding surface and electrical insulation properties of inter layer dielectrics for 3D monolithic integration. In *Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon*, pages 165–168, 2015.

[9] A. Koneru, S. Kannan, and K. Chakrabarty. Impact of electrostatic coupling and wafer-bonding defects on delay testing of monolithic 3D integrated circuits. *ACM Journal on Emerging Technologies in Computing Systems*, 13(4), 2017.

[10] E. J. Marinissen, T. L. McLaurin, and H. Jiao. IEEE std P1838: DfT standard-under-development for 2.5D-, 3D-, and 5.5D-SICs. In *IEEE European Test Symp,*, pages 1–10, 2016.

[11] Y. Xue, O. Poku, X. Li, and R. D. Blanton. PADRE: Physically-aware diagnostic resolution enhancement. In *IEEE Inter. Test Conf.*, pages 1–10, 2013.

[12] A. Koneru and K. Chakrabarty. An interlayer interconnect BIST and diagnosis solution for monolithic 3-D ICs. *IEEE Trans. CAD*, 39(10):3056–3066, 2020.

[13] A. Chaudhuri, S. Banerjee, and K. Chakrabarty. NodeRank: Observation-point insertion for fault localization in monolithic 3D ICs*. In *IEEE Asian Test Symp.*, pages 1–6, 2020.

[14] M. Joodaki. Uprising nano memories: Latest advances in monolithic three dimensional (3D) integrated Flash memories. *Microelectronic Engineering*, 164:75–87, 2016.

[15] Y. Yu and N. K. Jha. SPRING: A sparsity-aware reduced-precision monolithic 3D CNN accelerator architecture for training and inference. *IEEE Trans. Emerging Topics in Computing*, pages 1–1, 2020.

[16] A. Vandooren et al. 3D technologies for analog/RF applications. In *IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference*, pages 1–3, 2017.

[17] S.-C. Hung et al. Power supply noise-aware scan test pattern reshaping for at-speed delay fault testing of monolithic 3D ICs*. In *IEEE Asian Test Symp.*, pages 1–6, 2020.

[18] S.-Y. Huang. Chapter 7 - logic diagnosis. In *VLSI Test Principles and Architectures*, pages 397–459. Morgan Kaufmann, San Francisco, 2006.

[19] F. Scarselli et al. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[20] G. Zhang, H. He, and D. Katabi. Circuit-GNN: Graph neural networks for distributed circuit design. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proc. Inter. conf. Machine Learning*, volume 97 of *Proc. Machine Learning Research*, pages 7364–7373, 2019.

[21] Z. Xie et al. Net^2: A graph attention network method customized for pre-placement net length estimation. In *ASPDAC*, pages 671–677, 2021.

[22] Haralampos-G. Stratigopoulos. Machine learning applications in IC testing. In *IEEE European Test Symp.*, pages 1–10, 2018.

[23] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[24] J. Lee et al. Transfer learning for deep learning on graph-structured data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017.

[25] X. Han et al. Adaptive transfer learning on graph neural networks. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, page 565–574, 2021.

[26] B. W. Ku, K. Chang, and S. K. Lim. Compact-2D: A physical design methodology to build commercial-quality face-to-face-bonded 3D ICs. In *Inter. Symp. Physical Design*, pages 90–97, 2018.

[27] Y.-C. Lu et al. TP-GNN: a graph neural network framework for tier partitioning in monolithic 3D ICs. In *ACM/IEEE DAC*, pages 1–6, 2020.

[28] M. Wang et al. Deep graph library: Towards efficient and scalable deep learning on graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[29] J.P. Hayes. Digital simulation with multiple logic values. *IEEE Trans. CAD*, 5(2):274–283, 1986.

[30] A. Madkour et al. A survey of shortest-path algorithms. *CoRR*, abs/1705.02044, 2017.

[31] T.N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

[32] Z. Ying et al. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[33] J. Zhou et al. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[34] S. Panth et al. Placement-driven partitioning for congestion mitigation in monolithic 3D IC designs. *IEEE Trans. CAD*, 34(4):540–553, 2015.

[35] Y.-C. Lu et al. TP-GNN: A graph neural network framework for tier partitioning in monolithic 3D ICs. In *ACM/IEEE DAC*, pages 1–6, 2020.

[36] H. Abdi and L. J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[37] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proc. Inter. conf. Machine Learning*, page 233–240, 2006.

[38] N. V. Chawla et al. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[39] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000.

[40] C. Tan et al. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning*, pages 270–279, 2018.

Shao-Chun Hung received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 2019. He is currently pursuing Ph.D. in Electrical and Computer Engineering from Duke University, Durham, NC. He was an Intern with NVIDIA Corporation, Santa Clara, CA, and Cadence Design Systems, Austin, TX. His current research interests include reliability, testing, and diagnosis of monolithic 3D integrated circuits.

Sanmitra Banerjee received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, in 2018, and the M.S. and Ph.D. degrees from Duke University, Durham, NC, in 2021 and 2022, respectively. He is currently a Senior DFX Methodology Engineer at NVIDIA Corporation, Santa Clara, CA. His research interests include machine learning-based DFX techniques, and fault modeling and optimization of emerging AI accelerators under process variations and manufacturing defects.

Arjun Chaudhuri received a bachelor's degree in Electronics and Electrical Communication Engineering from the Indian Institute of Technology, Kharagpur, India, in 2017. He received a Ph.D. in Electrical and Computer Engineering from Duke University, USA, in 2022. He is currently a DFT Methodology engineer at NVIDIA, Santa Clara, USA. His research interests include design-for-testability and fault tolerance of machine learning hardware and monolithic 3D integrated circuits.

Jinwoo Kim received the B.S. degree and the M.S. degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea in 2011 and 2013. He is currently pursuing the Ph.D. degree in electrical and computer engineering, Georgia Institute of Technology, Atlanta, GA. He was an Analog Circuit Designer at Samsung Electronics, Hwaseong, South Korea from 2013 to 2017. His research interests include the design and EDA solutions for 2.5-D and 3-D ICs.

Sung Kyu Lim is a Professor of the School of Electrical and Computer Engineering, Georgia Institute of Technology. He received the B.S., M.S., and Ph.D. degrees from the Computer Science Department, University of California, Los Angeles (UCLA), in 1994, 1997, and 2000, respectively. He joined the School in 2001. His research focus is on the architecture, design, test, and electronic design automation (EDA) solutions for 2.5-D and 3-D ICs. His research is featured as Research Highlight in the Communication of the ACM in January, 2014. He is the author of Practical Problems in VLSI Physical Design Automation (Springer, 2008) and Design for High Performance, Low Power, and Reliable 3D Integrated Circuits (Springer, 2013). Dr. Lim has published more than 400 papers on 2.5-D and 3-D ICs.

Krishnendu Chakrabarty received the B. Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively. He is now the Fulton Professor of Microelectronics in the School of Electrical, Computer and Energy Engineering at Arizona State University (ASU). Before moving to ASU, he was the John Cocke Distinguished Professor and Chair of Electrical and Computer Engineering at Duke University. His current research projects include design-for-testability of 2.5D/3D integrated circuits, hardware security, failure prediction using AI/ML, AI accelerators, microfluidic biochips, AI for healthcare, and neuromorphic computing systems.