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

Shao-Chun Hung, Sanmitra Banerjee, Arjun Chaudhuri, and Krishnendu Chakrabarty
Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA

***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 39.19% improvement in diagnostic resolution with less than 1% loss of accuracy, compared to results from commercial tools.**

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. Fabricating upper-tier transistors in M3D designs with typical thermal budgets causes damage to wires and cells underneath [3]. 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 [4]. 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 [5]. 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 [6], which may lead to voids in MIVs during etching, resulting in delay defects and degradation of circuit performance [7]. 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 ascertain a known-good tier before assembly. Post-assembly methods such as [8] are not applicable to M3D due to large area overhead for wrapper cells around MIVs. In addition, delay-fault diagnosis catered to M3D designs is especially important as existing diagnosis methodologies cannot provide the high level of resolution (i.e., fault localization) needed at the tier level. 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). 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 ensure the compatibility of the proposed method with conventional scan-based designs and commercial tools, both with and without test compression.
- 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. We compare the effectiveness of our framework with a commercial fault-diagnosis tool in Section IV. In Section V, we discuss the transferability of our GNN model and propose a parameter tuning method to improve the diagnostic resolution. Finally, Section VI concludes the paper.

*This research was supported in part by the National Science Foundation under grant CCF-1908045.

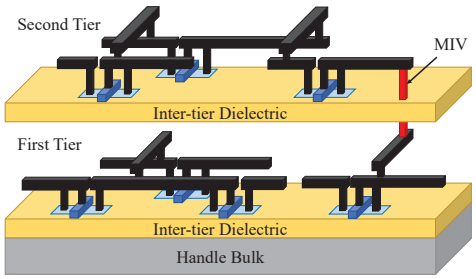


Fig. 1: Illustration of a two-tier M3D design.

II. Background

A. Monolithic 3D Integration

M3D integration processes active device tiers sequentially on a single wafer, as shown in Fig. 1. 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 [9]. Heterogeneous M3D architectures, i.e., multiple technology nodes for different tiers, have been explored recently to overcome physical limitations. In [10], heterogeneous M3D systems with different materials in each tier were predicted to be promising solutions for next-generation wireless communication.

While new applications of M3D integration have been explored, limited research efforts have been devoted to M3D testing and diagnosis. [11] developed an observation-point insertion algorithm for tier-level fault localization. A test pattern reshaping algorithm was proposed in [12] 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 [4] [5]. 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) [13]. 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.

Diagnostic resolution is reduced by test compression [14]. Typically, a response compactor is placed between scan chains and test output channels to help reduce test time and test data volume. This design increases the complexity of diagnosis due to the many-to-one mapping between scan chains and an output channel in the compactor. Scan flops that capture the erroneous responses can no longer be identified from the tester; therefore, known diagnosis methods for designs without compression are not usable in this scenario. Inserting bypass signals to access scan chains and generate uncompressed patterns for diagnosis can alleviate this problem [15]. However, on-line diagnosis based directly on test results requires uncompressed patterns, leading to an increased runtime. Techniques to conduct logic diagnosis with compressed patterns have also been proposed [16]; however, the diagnostic resolution in this way is lower than that for designs without compression. 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 gained 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 data such as images and text sequences. However, different graphs have different numbers of nodes/edges and irregular node connections. A preprocessing phase is therefore required to map graph structures to simplified representations, while topological dependency of each node may be lost during this phase [17].

Multiple tasks can be carried out using GNN-based structures. In node classification, the network labels nodes without ground-truth data by analyzing their neighbors. Link prediction evaluates the likelihood of an edge existing between two arbitrary nodes. Graph classification places graphs into different classes based on their structures and the features of nodes/edges. These tasks have been shown to be relevant and effective for solving various real-world problems [18]. Furthermore, the concept of ML is well suited to IC diagnosis because a large volume of data is collected throughout the production and product lifetime [19]. 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

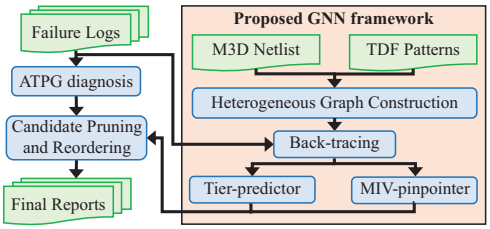


Fig. 2: GNN-based fault diagnosis flow.

(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.

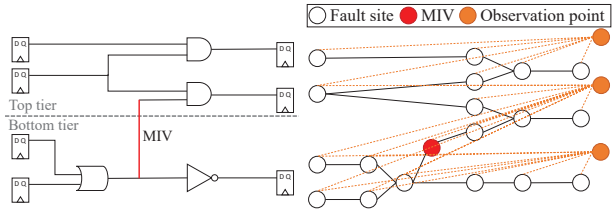
III. Proposed GNN-based Diagnosis Framework

In this section, we describe our GNN-based framework for tier-level fault diagnosis in M3D ICs. Fig. 2 presents a flowchart for the proposed diagnosis method. The first step is to convert the circuit under diagnosis (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 [20].

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. 3 illustrates the construction of our graph structure from a CUD. After graph construction, we apply ATPG patterns and conduct simulation



(a) Circuit netlist (b) Heterogeneous graph

Fig. 3: Illustration of the proposed heterogeneous graph structure.

```
   Input: Heterogeneous graph G, failure log file f from the tester
   Output: Subgraph G' ∈ 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. 4: Pseudo-code for the back-tracing algorithm.

with multiple logic values [21] to memorize transitions (i.e., whether a node switches from $0(1)$ to $1(0)$). We also utilize Dijkstra's Algorithm [22] to find the shortest path between both ends of a Topedge. The number of nodes and the number of MIVs in such shortest path establish the Topedge features.

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.

B. Back-tracing

Fig. 4 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.

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

Description	Type
Number of fan-in edges in the circuit	Numerical
Number of fanout edges in the circuit	Numerical
Number of Topedges connected	Numerical
Tier-level location	Binary
Level in topological order	Numerical
Whether it is a gate output	Binary
Whether it connects to an MIV	Binary
Number of fan-in edges in the sub-graph	Numerical
Number of fanout edges in the sub-graph	Numerical
Mean length of Topedges connected	Numerical
Standard deviation of length of Topedges connected	Numerical
Mean number of MIVs passed through by Topedges connected	Numerical
Standard deviation of number of MIVs passed through by Topedges connected	Numerical

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 leverage the graph convolutional network (GCN) [23] 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 I. In order to gather information and learn from neighbors of a node n, GCN layers are added to aggregate its features as follows [23]:

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

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. We formulate tier prediction as a graph classification problem in order to consider every candidate in the netlist. A graph pooling layer [18] 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. 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.

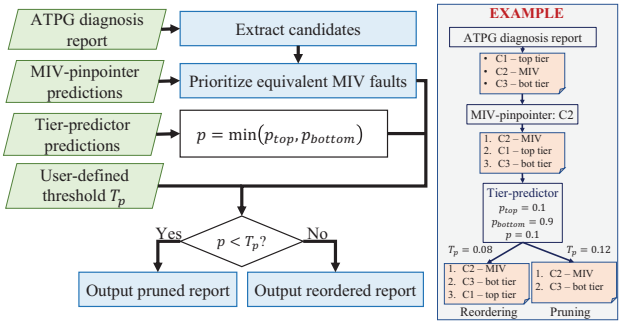


Fig. 5: Flowchart for candidate pruning and reordering.

TABLE II: 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%

D. ATPG Report Reordering and Pruning

Using the results from our Tier-predictor and MIV-pinpointer, we prune and reorder candidates in the ATPG diagnosis report to improve the diagnostic resolution and the FHI. Fig. 5 presents the flowchart for our pruning and reordering method. 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 [7], FHI can be improved in this way. Next, the minimum of $[p_{top}, p_{bottom}]$ is compared with a user-defined threshold value, T_p, to determine whether to prune or reorder candidates. If p_{top} (p_{bottom}) is smaller than the threshold, candidates in the top (bottom) tier are unlikely to be the ground-truth fault location. Such candidates are filtered out from the final candidate list. Otherwise, candidates are reordered by first appending fault sites in the tier predicted to be faulty, followed by those in the tier predicted to be fault-free. Note that filtering out candidates may occasionally lead to a loss of accuracy. However, when Tier-predictor points out the incorrect tier as faulty, the accuracy loss may be recovered by the MIV-pinpointer if the ground-truth fault happens to be equivalent to the MIV fault localized by the MIV-pinpointer. In the proposed solution, users can also fine-tune T_p to find the best trade-off between diagnostic resolution and accuracy.

IV. Experimental Results

A. Simulation Setup

We evaluated the proposed GNN-based diagnosis 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

TABLE III: Effectiveness of delay fault-localization in M3D benchmarks using ATPG diagnosis and the proposed GNN + ATPG framework.

Compaction mode	Design	ATPG diagnosis only (Commercial tool)		Proposed framework (GNN + ATPG)			
		Diagnostic resolution	Accuracy	Diagnostic resolution	Accuracy	FHI	T_p
Without compaction	AES	4.95	100%	4.20 (+15.15%)	99.07% (-0.93%)	1.85	0.08
	Tate	3.95	100%	3.34 (+15.64%)	99.07% (-0.93%)	1.37	0.08
	netcard	21.42	99.73%	13.03 (+39.19%)	99.20% (-0.53%)	8.44	0.08
	leon3mp	10.93	98.93%	8.10 (+25.81%)	98.40% (-0.53%)	3.66	0.01
With compaction	AES	9.36	97.47%	8.62 (+7.90%)	96.66% (-0.80%)	2.67	0.10
	Tate	5.31	99.87%	4.84 (+8.86%)	98.93% (-0.93%)	2.26	0.10
	netcard	23.17	98.93%	19.45 (+16.05%)	98.13% (-0.80%)	9.92	0.08
	leon3mp	12.91	98.13%	10.83 (+16.15%)	97.60% (-0.53%)	4.36	0.02

2012 benchmark suite. Each benchmark is synthesized with the open-source Nangate 45 nm standard cell library using Synopsys Design Compiler. Details about these M3D benchmarks are presented in [12]. Next, test-compression hardware is 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 $20\times$ in all benchmarks, that is, at most 20 scan chains are connected to one test output channel using the 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 II. To generate dataset for experiments, we randomly injected one TDF at a time in a circuit and carried out logic simulations with the generated patterns to obtain erroneous output responses. These responses were collected into a failure log file, which became a sample in the dataset. We generated 5000 samples for each benchmark, with and without response compaction, respectively.

B. Diagnosis with and without Response Compaction

We first examine the quality of our diagnosis framework using 750 samples (15% of the generated dataset) for each benchmark. We compare the average diagnostic resolution and accuracy of ATPG diagnosis reports among these samples with reports obtained after the reordering and pruning step. Table III shows the results for each benchmark, both with and without response compaction. For the results without compaction, the diagnostic resolution is improved by at least 15% with less than 1% loss of accuracy. This is significant because diagnosis reports are used to guide time-consuming physical silicon inspection. High diagnostic resolution minimizes the time spent on inspecting the fault-free parts of the chip.

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 III, the largest difference among four benchmarks is 8.39 for netcard without compaction; the size of the corresponding backup dictionary is only 246 kilobytes. Therefore, the memory overhead of the proposed method is within acceptable limits.

Note that the candidate fault sites in diagnosis reports generated by commercial ATPG tools are arranged alphabetically. As candidates cannot be prioritized for analysis, FHI is not applicable in such cases. In contrast, our framework prioritizes faults located in the predicted tier, as well as MIVs that are predicted to be faulty, in the final reports. The FHI is thus a good indicator to highlight the effectiveness of our algorithm. In three benchmarks, FHI is lower than half of the diagnostic resolution with the proposed framework, i.e., the ground-truth defect is in the first half of the candidate list in the final report.

Results for benchmark designs with response compaction are also shown in Table III. 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 16% improvement in diagnostic resolution with a very low accuracy loss. Furthermore, our approach does not require additional hardware or test data and is compatible with any combinational (e.g., XOR-based) response compactor.

V. Discussion

A. Transferability of Proposed GNN Framework

The transferability of ML enables pre-trained models to be applicable to new data without retraining. However, transferability is not feasible for our IC diagnosis problem because circuit and topological features (e.g., logic depth of circuit and the number of scan flops) are significantly different for different designs. We substantiate our claim by leveraging principle component analysis (PCA) [24] to visualize the distribution of the feature vectors listed in Table I. The visualization result for the Tate and leon3mp benchmarks (without response compaction) is shown in Fig. 6. Each sample represents a feature vector corresponding to a fault and the associated failure log. Clearly, features from the two benchmarks form separate clusters. As a result, a GNN framework trained on the dataset of one of these benchmarks cannot be directly transferred to perform diagnosis on the other benchmark. Moreover, training a single framework by using training data

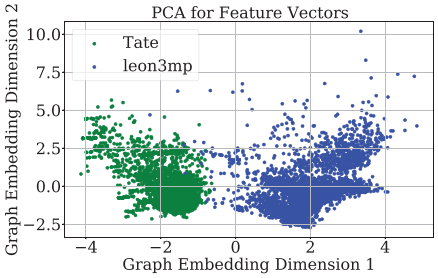


Fig. 6: Feature visualization of Tate and leon3mp benchmarks.

from both benchmarks may not converge due to the high intra-dataset variance that results from the gap in the two feature distributions. Therefore, it is recommended that the GNN-based Tier-predictor and MIV-pinpointer models are trained from scratch for different M3D designs.

B. Threshold value for candidate pruning and reordering

The threshold value, T_p, for candidate pruning and reordering is used to find the trade-off between diagnostic resolution and accuracy. In order to fine-tune T_p, we consider 15% of the samples for each benchmark as the validation dataset. We first make the predictions for samples in the validation dataset using the proposed Tier-predictor and MIV-pinpointer. Next, we carry out the candidate pruning and reordering process with different values of T_p to obtain the corresponding accuracy and diagnostic resolution. The distribution of accuracy and the improvement in diagnostic resolution for the netcard benchmark with and without response compaction is shown in Fig. 7. With this distribution, T_p can be fine-tuned to satisfy various objectives. For example, PFA requires a small value of diagnostic resolution to avoid high cost and runtime, which can be accomplished by choosing a large T_p. Our experiments aim at improving resolution with less than 1% loss of accuracy; therefore, we find the maximum T_p with which the loss of accuracy remains below 1%. Such values of T_p are used for the evaluations in Section IV. Results in Table III demonstrate that this fine-tuning method can achieve our goal of improving diagnostic resolution without violating the accuracy requirement.

VI. 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. The prediction results provide quick feedback to the foundry or diagnosis team prior to time-consuming failure analyses. We have also provided a candidate reordering and pruning algorithm based on our predictions to improve the quality of ATPG diagnosis reports. We have shown that with less than 1% loss of accuracy, diagnostic resolution is 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.

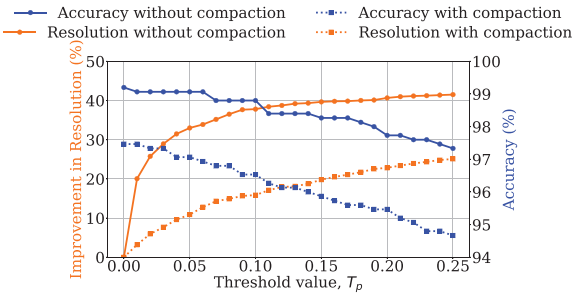


Fig. 7: Distribution of accuracy and the improvement in resolution among different threshold values for the netcard benchmark.

References

[1] S. Panth et al. Design challenges and solutions for ultra-high-density monolithic 3D ICs. In *S3S*, 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 JETCAS*, 2(4):714–722, 2012.

[3] A. Vandooren et al. Sequential 3D: Key integration challenges and opportunities for advanced semiconductor scaling. In *ICICDT*, pages 145–148, 2018.

[4] 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.

[5] M. Brocard et al. Impact of intermediate BEOL technology on standard cell performances of 3D VLSI. In *ESSDERC*, pages 218–221, 2016.

[6] K. Garidis et al. Characterization of bonding surface and electrical insulation properties of inter layer dielectrics for 3D monolithic integration. In *EUROSOI-ULIS*, pages 165–168, 2015.

[7] A. Koneru et al. Impact of electrostatic coupling and wafer-bonding defects on delay testing of monolithic 3D integrated circuits. *ACM JETC*, 13(4), 2017.

[8] E. J. Marinissen et al. IEEE Std P1838: DfT standard-under-development for 2.5D-, 3D-, and 5.5D-SICs. In *IEEE European Test Symp.*, 2016.

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

[10] A. Vandooren et al. 3D technologies for analog/RF applications. In *IEEE S3S*, pages 1–3, 2017.

[11] A. Chaudhuri et al. NodeRank: Observation-point insertion for fault localization in monolithic 3D ICs. In *IEEE Asian Test Symp.*, 2020.

[12] 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.*, 2020.

[13] L.-T. Wang et al. *VLSI Test Principles and Architectures: Design for Testability*. Elsevier, 2006.

[14] J. Rajski et al. X-press: Two-stage x-tolerant compactor with programmable selector. *IEEE Trans. CAD*, 27(1):147–159, 2008.

[15] J. Rajski et al. Embedded deterministic test for low cost manufacturing test. In *Proc. Int. Test Conf.*, pages 301–310, 2002.

[16] W.-T. Cheng et al. Compactor independent direct diagnosis. In *IEEE Asian Test Symp.*, 2004.

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

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

[19] H.-G. Stratigopoulos. Machine learning applications in IC testing. In *IEEE European Test Symp.*, 2018.

[20] 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.

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

[22] E. W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[23] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, 2016.

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