2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Test Smarter, Not Harder: GNN-Powered Automation for Post-Silicon Validation

Venkata Ajay Kolla
Venkata.ajay.kolla@intel.com

Abstract- Post-silicon validation presents significant challenges in modern semiconductor development, particularly in managing extensive test suites across multiple domains and SKUs. Traditional approaches execute thousands of tests indiscriminately, leading to substantial inefficiencies and resource waste. Manual test selection relies heavily on domain expertise, which is both error-prone and time-consuming. This paper introduces a novel Graph Neural Network (GNN) based framework that automates intelligent test selection by performing similarity analysis on test metadata including intent descriptions, categorical attributes, and historical dependencies. This approach leverages Graph Attention Networks (GAT) to create meaningful test embeddings that enable efficient clustering and selection of representative tests. Preliminary simulation results demonstrate that the proposed metadata-driven approach achieves F1-score of 0.8 when compared to expert-generated test groupings, while reducing test execution overhead by 30-50% and maintaining 90 coverage efficiency.

Keywords - Post-silicon validation (PSV), Graph Neural Networks (GNNs), Graph Attention Networks (GATs), Test suite optimization, Machine Learning (ML), System-on-Chip (SoC), Metadata-driven test selection, Stock Keeping Units(SKUs), Bidirectional Encoder Representations from Transformers (BERT)

I. INTRODUCTION

The semiconductor industry faces mounting pressure to reduce time-to-market while maintaining rigorous validation standards for increasingly complex system-on-chip (SoC) designs. Post-silicon validation, which is the process of testing fabricated semiconductor chips to verify their functionality, performance, and compliance with design specifications. represents a critical bottleneck in the development cycle. Unlike pre-silicon validation that operates on simulation models, post-silicon validation must contend with real hardware constraints, limited observability, and the substantial cost of fabricated devices.
Modern validation teams typically manage test suites containing thousands of individual tests spanning multiple domains including CPU functionality, memory subsystems, I/O interfaces, power management, and thermal controls. These must be executed across numerous SKUs and configuration combinations, creating validation matrices that easily expand to tens of thousands of executions per cycle. Running all available tests ensures thoroughness but introduces critical inefficiencies: equipment is tied up executing redundant tests, engineering effort is diverted to managing bloated suites, and delayed feedback slows design iteration.

Traditionally, validation teams mitigate this by relying on expert-driven test selection. Engineers manually curate subsets they believe best represent the validation goals of a given cycle. While this leverages valuable domain expertise, it is subjective, inconsistent, and time-consuming; often stretching to weeks for large-scale suites. As teams evolve, the original rationale behind test intent is frequently lost, fragmenting institutional knowledge and further compounding inefficiency. Existing automated methods in post-silicon validation focus largely on runtime monitoring and stress testing, which require extensive data collection from hardware during execution. These techniques, while effective in specialized scenarios, fail to address the fundamental question: ***which tests should be selected in the first place, and why?*** An intelligent, scalable approach to test selection remains largely absent in current industrial practice.

This work introduces such an approach where a Graph Neural Network (GNN)-based framework is introduced that models the validation test space as a heterogeneous graph wherein nodes represent tests and edges encode relationships derived from metadata such as textual descriptions, categorical attributes, and historical dependencies. Using a Graph Attention Network (GAT), the framework automatically learns which relationships matter most, enabling effective

clustering and selection of representative tests. Unlike runtime monitoring solutions, this approach is metadata- only, i.e., it requires no performance counters, no real-time collection, and no manual intervention.
The contributions of this paper are as follows:

- First application of GNNs to post-silicon test selection, demonstrating how graph learning can capture latent structure in validation suites.
- A metadata-driven methodology that eliminates dependence on expensive runtime data or hardware instrumentation.
- A scalable automated framework that reduces human effort while maintaining or improving coverage compared to expert-defined selection.
- Validation on representative industrial test subsets, showing significant improvements in efficiency and coverage, with promising scalability to full production environments.

II. RELATED WORK

The application of machine learning techniques to hardware validation represents an emerging research area with limited prior work specifically addressing automated test selection in post-silicon environments. This section examines relevant research in both machine learning for hardware testing and graph neural network applications that inform our approach.

Most prior work has concentrated on pre-silicon verification. Chen *et al.* [4] used supervised learning for test case prioritization in RTL verification, achieving improved bug detection but requiring extensive labeled data and operating only in simulation. Hu and Liu [5] applied clustering techniques for test suite reduction, demonstrating reduced execution time while preserving fault detection capability. However, their reliance on code coverage metrics and focus on simulation limit applicability to post-silicon environments. In parallel, advances in graph neural networks (GNNs) have shown promise for structured data analysis and similarity-driven tasks. Hamilton *et al.* [1] introduced GraphSAGE, enabling inductive learning on unseen nodes which is relevant in validation settings where new tests are continuously added. Veličković *et al.* [2] proposed Graph Attention Networks (GAT), which dynamically weight the importance of node relationships, a capability well suited for heterogeneous test metadata. Ying *et al.* [3] presented PinSage, a GNN for large-scale recommendation systems, illustrating the effectiveness of graph-based similarity search in massive heterogeneous datasets.

Despite these advances, significant gaps remain in applying machine learning to post-silicon validation challenges. Existing approaches in hardware validation typically require extensive labeled training data or rely on metrics that are difficult to obtain in post-silicon environments. Performance-based approaches, while intuitive, introduce complexity and overhead that may not be justified for test selection applications. Most critically, prior work has not addressed the unique characteristics of post-silicon validation environments: limited observability, heterogeneous test types, legacy test suites with incomplete documentation, and the need for rapid test selection decisions. The proposed work attempts to address these gaps by developing a framework specifically designed for the constraints and requirements of post-silicon validation.

III. METHODOLOGY

Problem Formulation

The test selection problem is formulated as a **graph-based similarity search** over test metadata. Consider a validation test corpus:

$$T = \{t_1, t_2, \ldots, t_n\}$$

with associated metadata

$$M = \{m_1, m_2, \ldots, m_n\}$$

The objective is to learn **embeddings**:

$$E = \{e_1, e_2, \ldots, e_n\}$$

such that each embedding captures both semantic and functional similarities between tests. These embeddings enable clustering and selection of representative subsets that maintain coverage while minimizing execution overhead. The dimensional reduction from 768 (BERT) to 128 dimensions serves multiple purposes: (1) computational efficiency during graph operations, as GAT layers scale quadratically with embedding size, (2) prevention of the curse of dimensionality in similarity computations, and (3) creation of a unified embedding space that balances textual semantics with categorical and dependency features. The 128-dimensional target was chosen as it provides sufficient representational capacity while remaining computationally tractable for real-time inference in production validation environments.

Graph Construction

The validation corpus is represented as a heterogeneous graph:

$$G = (V, E)$$

- **Nodes (V):** represent individual tests.
- **Edges (E):** represent relationships between tests.

This graph structure allows modeling of multidimensional relationships not captured by traditional selection techniques.

1) Node Features

Each test node aggregates three categories of features:

- **Textual Embeddings:** Generated using **BERT**, providing 768-dimensional semantic representations of test descriptions. These embeddings capture meaning beyond keywords and allow grouping of functionally similar tests written in different terminologies.
- **Categorical Features:** One-hot encodings of discrete attributes such as application domain (CPU, memory, I/O, power), test type (functional, stress, regression), environment constraints, and SKU configurations. These ensure structural grouping across domains.
- **Dependency Features:** Binary vectors encoding historical co-execution and dependency information. This leverages accumulated validation knowledge that is not explicitly recorded in documentation.

2) Edge Construction

Edges encode similarity across multiple dimensions:

- **Semantic Similarity:** Cosine similarity of BERT embeddings; edges are formed when similarity > τ=0.7. The similarity threshold τ=0.7 was selected based on empirical analysis of BERT embedding distributions in our validation corpus. Thresholds below 0.6 resulted in overly dense graphs with weak semantic connections, while thresholds above 0.8 created sparse graphs that failed to capture meaningful test relationships. The value of 0.7 strikes an optimal balance.
- **Categorical Overlap:** Weighted edges represent the degree of attribute overlap (e.g., same domain or SKU).
- **Historical Co-execution:** Edge weights proportional to co-occurrence frequency in past validation campaigns.

Graph Neural Network Architecture

The architecture (Fig. 1) is based on Graph Attention Networks (GAT), chosen for their ability to dynamically weight different relationships. Unlike Graph Convolutional Networks (GCNs), GATs can assign higher importance to specific neighbors, which is crucial when some metadata links are more relevant than others. The following subsections go deeper on the architecture:

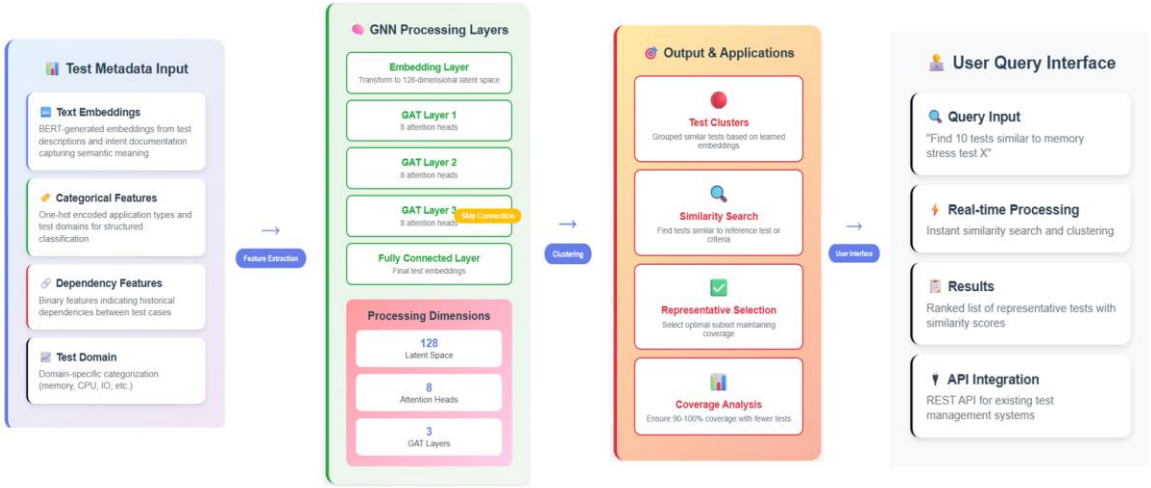

Fig. 1. Graph Neural Network architecture showing the processing pipeline from test metadata input through GAT layers to final test embeddings

1) Embedding Layer

Raw node features are concatenated and linearly transformed Let h_i denote the hidden representation of token i, formed by concatenating the BERT embedding, categorical features, and dependency features, followed by a linear transformation

$$h_i^0 = W_0 \cdot [\, BERT(t_i) \parallel categorical(t_i) \parallel dependency(t_i)] + b_0$$

where W_0 , b_0 are learnable parameters, and ‖ denotes concatenation. This projects all features into a 128-dimensional latent space.

2) Graph Attention Layers

Three GAT layers are applied sequentially. Each layer employs 8 attention heads to aggregate features:

$$h_i^{l+1} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij}^l \cdot W^l h_j^l \right)$$

where:

- N(i) = neighbors of node i,
- α_{ij}^l = attention weight assigned to neighbor jjj,

- σ(·) = non-linear activation.

This mechanism enables context-aware weighting of test relationships.

3) Skip Connections

To prevent over-smoothing (where deeper GNNs make embeddings indistinguishable), a skip connection is added:

$$h_i^{final} = h_i^2 + W_{skip} \cdot h_i^0$$

This retains original node information while benefiting from learned relational features.

4) Output Layer

The final fully connected layer outputs normalized embeddings:

$$e_i = normalize\left(W_{out} \cdot h_i^{final} + b_{out}\right)$$

The normalize () function applies L2 normalization, ensuring all test embeddings lie on the unit hypersphere. This enables efficient cosine similarity computation for clustering and selection, as normalized embeddings allow cosine similarity to be computed as simple dot products. These final embeddings are used for similarity search and clustering.

Training Parameters

- **Loss function:** Training is based on contrastive loss, which enforces that similar tests have embeddings close together, while dissimilar tests are far apart
- **Optimizer**: Adam is selected for its adaptive learning rate properties, which handle sparse gradients effectively, a common scenario in graph-based learning.
- **Learning Rate**: Set to 0.001, based on prior GNN studies [1], [2], balancing convergence speed and stability.
- **Weight Decay**: 1e-4 to prevent overfitting by penalizing large weights.
- **Cross-validation**: 5-fold cross-validation ensures robustness.
- **Early Stopping**: Applied based on validation loss to avoid overfitting and reduce training time.

This structured methodology ensures that the learned embeddings effectively capture semantic, categorical, and historical aspects of validation tests while remaining scalable to large industrial datasets.

IV. IMPLEMENTATION

The implementation follows a structured pipeline, illustrated in Fig. 2, progressing from data preprocessing to deployment within validation workflows. Each stage is designed for scalability and practical applicability in industrial test environments.

Data Preprocessing

Test metadata is aggregated from multiple validation sources, including documentation repositories, execution logs, and configuration management systems. A preprocessing pipeline standardizes formats, addresses missing or incomplete information, and produces structured feature representations.

- Text Processing: Test descriptions and intent documentation are normalized (lowercasing, punctuation removal, and stop word filtering). Semantic representations are extracted using the pre-trained *bert-base-uncased* model, resulting in 768-dimensional embeddings per test description.
- Categorical Encoding: Metadata such as application domains, test types, and SKU configurations is one-hot encoded into binary feature vectors. A dedicated "unknown" category is used to represent missing values while preserving structural integrity.
- Dependency Analysis: Historical execution logs are analyzed to extract temporal and co-occurrence dependencies. Features include execution order, shared resource usage, and correlated failures, forming dependency vectors that capture relational characteristics among tests.

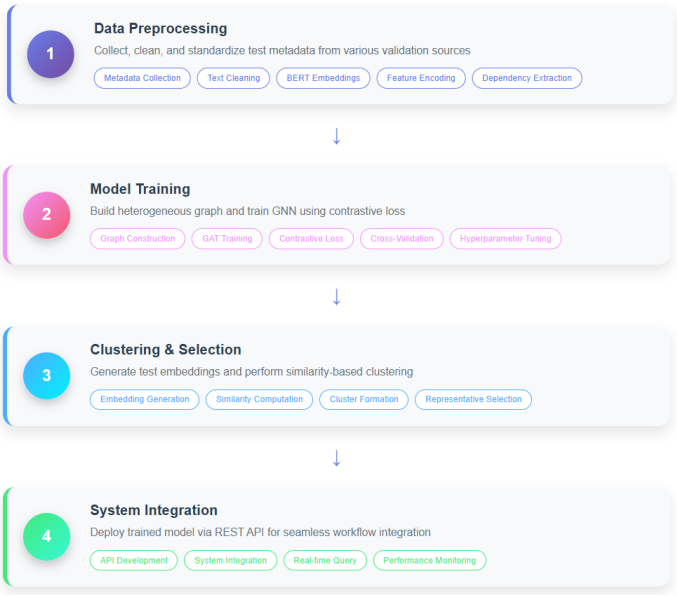

Fig. 2. Implementation workflow showing the four-phase process

Model Training

The graph neural network is implemented using PyTorch Geometric, leveraging GPU acceleration for training and CPU-based inference for deployment. Graph Attention Networks (GATs) are employed to weigh different relationships dynamically, allowing the model to focus on more influential connections. The training uses a contrastive learning objective, encouraging embeddings of similar tests to be closer while dissimilar tests are pushed apart.

The model begins by processing heterogeneous input features from multiple sources. Test descriptions are encoded using the pre-trained bert-base-uncased model, producing 768-dimensional semantic embeddings that capture textual intent and functionality. Categorical metadata (domains, test types, SKU configurations) contributes 20-50 dimensions through one-hot encoding, while dependency features derived from historical execution patterns add another 10-30 dimensions. These multi-modal features are concatenated, resulting in combined feature vectors of approximately 800-850 dimensions per test.

This high-dimensional concatenated input undergoes dimensional reduction through the initial embedding layer, projecting all features into a unified 128-dimensional latent space. This reduction serves multiple critical purposes: (1) computational efficiency, as Graph Attention Network operations scale quadratically with embedding size, making 800+ dimensions computationally prohibitive for real-time inference in production environments; (2) feature

harmonization, ensuring textual, categorical, and dependency information contribute proportionally rather than being dominated by the dense BERT representations; and (3) regularization, preventing overfitting to high-dimensional semantic features while preserving essential relational information.
The 128-dimensional target was selected through empirical analysis balancing representational capacity with deployment constraints. Dimensions below 64 resulted in significant information loss, particularly in semantic similarity detection, while dimensions above 256 provided marginal improvements at considerable computational cost.

Clustering & Selection:

Once embeddings are generated, similarity-based clustering identifies representative subsets of tests.

- **Clustering Quality:** Agreement with **expert-defined clusters** (domain-specific groupings created by validation engineers) is quantified using the **F1-score** [7]. This metric balances precision and recall, ensuring selected clusters capture relevant tests without excessive redundancy.
- **Coverage Analysis:** Selected subsets are benchmarked against the full test suite, with measurement of functional coverage
- **Representative Selection:** From each cluster, a minimal set of tests is chosen that maximizes representativeness while minimizing execution overhead.

System Integration

For deployment, the trained model is proposed to be exposed through a REST API that integrates directly with test management workflows.

- **API Capabilities:** Support similarity queries, cluster-based selection, and specification of custom validation objectives.
- **Workflow Integration:** Engineers should be able to invoke the model directly from existing tools without altering existing validation processes.
- **Performance Monitoring:** Real-time metrics which track model accuracy, coverage, and efficiency in production environments, ensuring continuous reliability.

V. RESULTS

TABLE 1
PERFORMANCE COMPARISON BETWEEN BASELINE AND PROPOSED APPROACH

Approach	Functional Coverage (%)	Test Reduction (%)	Selection Time	F1 score
Expert-Defined Grouping(baseline)	95	–	Several days	–
Automated Framework (GAT)	90	30-50	< 10 minutes	0.8

The evaluation was conducted on a subset of 100 validation tests drawn from representative post-silicon domains. These test cases include a mix of protocol, driver, and functional checks, representative of real-world post-silicon validation scenarios. For baseline comparison, expert-defined groupings were used. These groupings represent the manual categorization of tests typically done by senior validation engineers based on functional intent, coverage requirements, and prior domain knowledge.

The proposed framework was implemented in PyTorch and trained on test metadata (intent descriptions, categorical attributes, and dependency features). A Graph Attention Network (GAT) was employed to learn embeddings for clustering and selection.

As shown in Table 1, the automated framework achieved:

- Coverage comparable to expert-defined grouping (90% vs. 95%)
- Test reduction ranging from 30% to 50% depending on the domain
- F1-score of 0.8, closely matching the range reported in prior ML-based test selection approaches [6].
- Reduced test curation time from weeks of manual effort to minutes

Overall validation execution time is reduced by 40-60% due to the combination of fewer test executions and automated selection processes. This time savings directly translates to faster feedback cycles and reduced laboratory resource requirements. While the results indicate promising efficiency gains on this limited dataset, larger-scale evaluation across industrial workloads will be required to confirm generalization.

VI. Conclusion

This work presents a Graph Neural Network based framework for intelligent test selection in post-silicon validation, leveraging only test metadata such as intent descriptions, categorical attributes, and historical dependencies. By moving away from manual expert-driven clustering and performance-monitoring–based selection, the proposed approach achieves **30–50% reduction in test execution overhead** and an **F1-score of 0.8** compared to expert-defined groupings, while maintaining **near-equivalent coverage efficiency**. These improvements directly translate into tangible cost savings by lowering compute resource usage, reducing engineering effort in manual selection, and accelerating product bring-up timelines.

It should be noted that this study combines theoretical formulation with preliminary implementation, and the current evaluation is limited to smaller-scale datasets. Scaling the approach to industrial-scale validation datasets is an essential next step to fully establish robustness and generalizability. Nevertheless, the findings highlight the promise of metadata-based GNNs as a viable path toward reducing reliance on costly manual selection while maintaining validation quality.

References

[1] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," International Conference on Learning Representations (ICLR), 2018.

[3] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pp. 974–983, 2018.

[4] L. Chen, M. A. Hsiao, and T. Wang, "Machine learning guided test case prioritization for hardware verification," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 11, pp. 2016-2057, Nov. 2019.

[5] Y. Hu and X. Liu, "Clustering-based test suite reduction for hardware verification," Design Automation Conference (DAC), pp. 1-6, 2020.

[6] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" International Conference on Learning Representations (ICLR), 2019

[7] Powers, D. M. W., "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," JMLT, 2011