

SubNM-Technology-Node-Aware Hardware Trojan Detection via Multi-Library Netlist Synthesis and Modular Graph Neural Networks

Anindita Chattopadhyay

BMS College of Engineering

Siddharth Bisariya

Aeronautical Development Establishment

Vijay Kumar Sutrar

`vks.ade@gov.in`

Aeronautical Development Establishment

Research Article

Keywords: Hardware Trojans, Graph Neural Networks, Technology Node Synthesis, Modular GNN, RTL Security, Trust-Hub, Cross-Node Generalization

Posted Date: August 6th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10361560/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

SubNM- Technology-Node-Aware Hardware Trojan Detection via Multi-Library Netlist Synthesis and Modular Graph Neural Networks

Anindita Chattopadhyay^{1*}, Siddharth Bisariya^{2*}, Vijay Kumar Sutrakar²

Abstract

The growing complexity of modern system-on-chip (SoC) architectures has intensified the threat of Hardware Trojans (HTs), which can maliciously alter functionality, leak sensitive data, or enable remote activation post-fabrication. While graph neural network (GNN), based detection frameworks have demonstrated strong performance on Register Transfer Level (RTL) designs, they are consistently evaluated under a single synthesis configuration, ignoring the profound structural variability that technology-node scaling introduces at the gate level. In the present work, a technology-node-aware HT detection framework, i.e. SubNM is proposed that systematically synthesizes the same 21 Trust-Hub RTL benchmark designs across four open-source standard cell libraries, (i) OSU180 (180 nm), (ii) TSMC90 (90 nm), (iii) NanGate45 (45 nm), and (iv) SAED28 (28 nm) using the Yosys open-source synthesis suite, thereby generating an 84-design multi-node corpus. Each synthesized netlist is hierarchically partitioned into semantically coherent graphs, which are processed by a GNN employing joint structural and technology-aware node embeddings augmented with a library one-hot encoding and normalized gate area. Cross-node evaluation protocol is introduced to measure generalization by training all the libraries and testing on single library file. Experiments demonstrate that individual libraries show average (min) TPR of 93.63%, 89.51%, 85.12 and 78.68% for OSU180, TSMC90, NanGate45 and SAED28. However, after utilization of SubNM, the performance improves to 99.46%, 99.12%, 98.98% and 98.84 for OSU180, TSMC90, NanGate45 and SAED28. These results establish SubNM as a scalable, library-agnostic solution for pre-silicon hardware security assurance across the full contemporary technology spectrum.

Keywords: *Hardware Trojans, Graph Neural Networks, Technology Node Synthesis, Modular GNN, RTL Security, Trust-Hub, Cross-Node Generalization.*

²Vijay Kumar Sutrakar, vks.ade@gov.in | ¹Department of Electronics and Communication, BMS College of Engineering, Bengaluru, Karnataka, India. ²Aeronautical Development Establishment, Defence Research and Development Organisation, Bengaluru, Karnataka, India. | *Authors have equally contributed.

1 Introduction

The globalization of the semiconductor supply chain has made integrated circuit (IC) designs vulnerable to malicious modifications at multiple pipeline stages, including IP integration, logic synthesis, and fabrication [1]. HTs deliberately inserted alterations that can leak sensitive information, alter functional behaviour, or enable covert backdoor activation represent one of the most critical threats in modern SoC security [2, 3]. The stealth and diversity of HT insertion strategies demand detection frameworks that remain effective not only across design families but also across the varying structural forms a design assumes when synthesized for different technology nodes.

Modern semiconductor products are routinely re-targeted across technology nodes, i.e., an IP block originally developed for a 180 nm process is often re-synthesized for 90 nm, 45 nm, or 28 nm libraries to meet area, power, or performance targets in successive product generations. Each re-synthesis produces a structurally distinct netlist the gate count, fan-in/fan-out distributions, and logic depth change significantly with the standard cell library even though the RTL specification and any embedded Trojans remain unchanged [4]. This technology-induced structural variability is a blind spot for existing GNN-based HT detection frameworks, which are trained and evaluated on a single synthesis configuration. A detector that cannot generalize across technology nodes is a fragile security primitive in multi-generation supply chains.

The recently proposed SubRTL framework [5] addresses the scalability challenge of HT detection by decomposing large RTL designs into modular subgraphs and performing parallel hierarchical GNN inference, achieving state-of-the-art detection accuracy with up to $5\times$ memory reduction and $3\times$ inference speedup compared to full-graph GNN baselines. However, SubRTL like its predecessors is evaluated exclusively on RTL designs synthesized under a single library configuration, leaving open the question of how the learned Trojan representations transfer when the same design is compiled to a different technology node.

This paper directly addresses this gap by introducing SubNM, a technology-node-aware extension of the SubRTL methodology. The core contribution is the construction of a multi-node benchmark corpus, the same 21 Trust-Hub RTL designs used in SubRTL are synthesized across four representative standard cell libraries OSU180, TSMC90, NanGate45, and SAED28 using Yosys [6], producing 84 uniquely structured netlists that span legacy and advanced technology nodes. Each synthesized netlist is processed through the identical SubRTL pipeline, hierarchical parsing, modular subgraph partitioning, node feature extraction with technology-aware attributes, and parallel GNN inference using the AdaBoB optimizer. Initially 21 Trust Hub

RTL designs are evaluated on individual standard cell libraries separately. This study is carried out to understand how the different cell libraries and their architecture behaves against a particular cell library when their details are not intended. In order to overcome the degraded performance of individual cell, a cross-node generalization protocol is proposed and evaluated, in which the model is trained on netlists from one library family and tested on netlists synthesized with a different library, directly measuring the transferability of learned Trojan representations under library-induced structural drift.

The remainder of this paper is organized as follows. Section 2 reviews related work on HT detection and technology-aware hardware analysis. Section 3 details the methodology, including the multi-library synthesis flow, graph construction with technology-aware features, and the cross-node evaluation protocol. Section 4 describes the computational setup. Section 5 presents the results and analysis. Section 6 concludes the paper.

2 Related Work

HT detection research has evolved through four broad paradigms such as side-channel analysis, functional simulation and formal verification, structural netlist inspection, and graph learning. A comprehensive review of these paradigms can be found in [7, 8]. The present work positions itself at the frontier of graph learning-based detection, extending it with explicit multi-node synthesis awareness.

Early HT detection techniques exploited differences in power transients, electromagnetic signatures, or timing delay profiles between golden and suspect chips [9, 10]. While effective for small circuits, these methods require a trusted golden reference, suffer from process variation at advanced nodes where Trojan-induced differences are masked by natural parameter spread, and cannot localize the Trojan at the gate level [11]. Parametric Trojans further evade side-channel analysis by minimizing switching activity, making these approaches unreliable for stealthy threats at 45 nm and below [12].

Structural methods analyze gate-level properties such as fan-in, fan-out, path rarity, and controllability/observability (COTD) to distinguish malicious from benign regions [13, 14]. These approaches do not require a golden reference but typically treat all gates uniformly, failing to exploit functional context, and exhibit high false-positive rates in structurally dense regions. It also does not address technology-node variability, a feature threshold calibrated on a 180 nm netlist will be miscalibrated when applied to a 45 nm synthesis of the same design.

The natural representation of a circuit as a directed graph of logic gates and signal connections has motivated extensive use of GNNs for HT detection [15-18]. SubG4TJ [19] constructs multi-dimensional subgraph representations combining node-level behavioural features with structural connectivity. GREAT [20] employs global representation with edge attention for fine-grained localization. SubRTL [5] introduced modular, parallel GNN inference on hierarchically partitioned RTL graphs, achieving state-of-the-art TPR with dramatic memory and runtime reductions. Despite these advances, no existing work evaluates GNN-based detectors under a multi-library synthesis regime or explicitly measures cross-node generalization, which is the central contribution of the present paper.

Technology mapping during logic synthesis is a well-studied topic in EDA, with Yosys [6] being widely used for open-source synthesis flows. The structural impact of technology mapping on security properties has received limited attention. A key observation is that the same RTL, when compiled to different standard cell libraries, yields netlists with markedly different structural statistics, cell counts, interconnect density, and logic depth vary by $2\text{-}5\times$ between a 180 nm and a 28 nm library [21]. This implies that detection models trained on single-node netlists are implicitly over-fitted to the structural biases of a particular library family. SubNM is the first framework to explicitly quantify and mitigate this limitation.

3 Methodology

The SubNM pipeline extends SubRTL with two additional stages as shown in Figure 1, multi-library synthesis and technology-aware feature augmentation, while preserving the modular GNN inference architecture. The overall flow consists of five stages: (a) RTL dataset collection, (b) multi-library synthesis using Yosys, (c) hierarchical parsing, (d) technology-aware graph construction, and (e) GNN inference.

3.1 Dataset Collection

The SubNM dataset consists of 21 Trust-Hub RTL benchmark designs. These designs include combinational circuits, large sequential circuits, and the industrial-scale EthernetMAC10GE design. For cross training protocol the model is tested with unseen circuits (c2670_T010, c2670_T014, PIC16F84-T100, PIC16F84-T200). Each design is synthesized with all four libraries, yielding 84 benchmark-library instances.

Each of the 21 Trust-Hub RTL benchmark designs is synthesized using Yosys with four representative open-source standard cell libraries that span the commercially relevant technology spectrum from legacy to advanced nodes:

- OSU180 (180 nm): A standard-cell library developed at Oklahoma State University for educational and research use, representative of legacy fabrication nodes [22].
- TSMC90 (90 nm): A standard-cell library targeting the 90 nm technology generation, representing the transition from micron-scale to deep sub-micron processes [23].
- NanGate45 (45 nm): The NanGate Open Cell Library based on a predictive 45 nm PDK, widely used as a benchmark for logic synthesis research [24].
- SAED28 (28 nm): The Synopsys Academic Educational Design (SAED) library at 28 nm, representing advanced near-modern process technology [25].

3.2 Multi-Library Synthesis Using Yosys

For each RTL design, the Yosys synthesis flow proceeds in three steps: (i) elaboration with the `read_verilog` and `hierarchy` commands to parse and elaborate the design hierarchy; (ii) logic optimization using `opt`, `techmap`, and `flatten` to simplify and normalize the logic; and (iii) technology mapping via `abc -liberty` with the target cell library to produce a gate-level netlist. The resulting 84 netlists (21 designs $\times$ 4 libraries) form the SubNM benchmark corpus. Each netlist is exported in Verilog format and subsequently processed by the SubRTL hierarchical parsing stage.

Synthesis is performed using Yosys 0.38 with the following script template for each design (D) and library (L):

```
read_verilog -sv design_D.v
hierarchy -check -top top_module
proc; opt; fsm; opt; memory; opt
techmap; opt
abc -liberty lib_L.lib; clean

yosys -p "read_verilog design.v; synth -top top; abc -liberty osu180.lib;
write_verilog design_180.v"
yosys -p "read_verilog design.v; synth -top top; abc -liberty nangate45.lib;
write_verilog design_45.v"
yosys -p "read_verilog design.v; synth -top top; abc -liberty saed28.lib;
write_verilog design_28.v"
```

3.3 Technology-dependent Gate Level Netlist

Each graph $G_i = (V_i, E_i)$ is constructed as a directed graph where nodes represent logic gates and edges represent signal flow. The node feature vector $\mathbf{x}_i \in \mathbb{R}^d$ is augmented with technology-aware attributes beyond the structural features used in SubRTL.

3.4 Graph construction and node features

The complete feature set includes:

- One-hot encoded gate type (standardized across all four libraries to a common cell taxonomy comprising INV, BUF, AND, OR, NAND, NOR, XOR, XNOR, MUX, DFF, and their drive-strength variants).
- Fan-in and fan-out counts.
- Logic depth index from primary inputs.
- Hierarchical module tag.
- Technology node encoding: a 4-dimensional one-hot vector identifying the synthesis library (OSU180, TSMC90, NanGate45, SAED28). This encoding explicitly informs the GNN of the fabrication context, enabling the model to learn library-conditioned Trojan signatures.
- Normalized gate area: the cell area (in μm^2) from the standard cell library, normalized within the subgraph. This feature captures the physical footprint differentials that distinguish libraries and correlates with switching activity patterns relevant to payload-type Trojans.

All features are normalized to zero mean and unit variance within each subgraph. The adjacency matrix A encodes directed signal connections, and edge attributes encode fan-out multiplicity. The full feature dimension is $d = 24$ for the extended technology-aware configuration.

3.5 GNN Architecture

The SubNM framework employs a modular Graph Neural Network (GNN) architecture for node-level Hardware Trojan detection on gate-level netlist subgraphs. Each node is associated with a 24-dimensional feature vector that integrates both structural attributes, such as gate type, fan-in/fan-out, and logic depth and technology-aware features, including module identifiers, technology-node encoding, and normalized gate

area. This feature formulation enables the model to capture both logical behaviour and fabrication-context information.

The core of the architecture consists of three stacked graph convolutional layers with dimensions $6424 \rightarrow 64 \rightarrow 128 \rightarrow 64$. These layers perform iterative message passing, where node embeddings are updated by aggregating information from their local neighbourhoods according to $h_v^{(k)} = \sigma \left(W^{(k)} \cdot \text{AGG}(\{h_u^{(k-1)} : u \in N(v)\}) \right)$, $h_v^{(k)}$ denotes the embedding of node v at layer k , $N(v)$ represents its neighboring nodes, $W^{(k)}$ is a learnable weight matrix, and σ is the ReLU activation function. Through successive layers, the model captures increasingly higher-order structural dependencies, enabling it to learn complex patterns associated with Trojan behaviour.

To further enhance representation learning, an attention mechanism is incorporated into the aggregation process. This mechanism assigns adaptive importance weights to neighbouring nodes, computed as $\alpha_{vu} = \frac{\exp e_{vu}}{\sum_{k \in N(v)} \exp e_{vk}}$, allowing the model to prioritize structurally significant nodes while suppressing less relevant connections. This is particularly important for detecting stealthy Trojans, which often manifest as subtle and sparsely connected anomalies within the circuit.

Following feature propagation, the learned node embeddings are aggregated using global mean pooling to obtain a subgraph-level representation, defined as $h_G = \frac{1}{|V|} \sum_{v \in V} h_v$. This pooled representation captures the overall structural characteristics of the subgraph and serves as input to the classification stage. A fully connected layer with sigmoid activation is then applied to compute the probability of Trojan presence, given by $\hat{y} = \sigma(W h_G + b)$.

In addition to subgraph-level classification, the node embeddings are also utilized for fine-grained localization of Trojan nodes, enabling the framework to identify malicious regions within the circuit. This combination of multi-layer graph convolution, attention-based aggregation, and technology-aware feature integration allows the SubNM architecture to effectively capture both local and global structural patterns while maintaining robustness across different technology nodes.

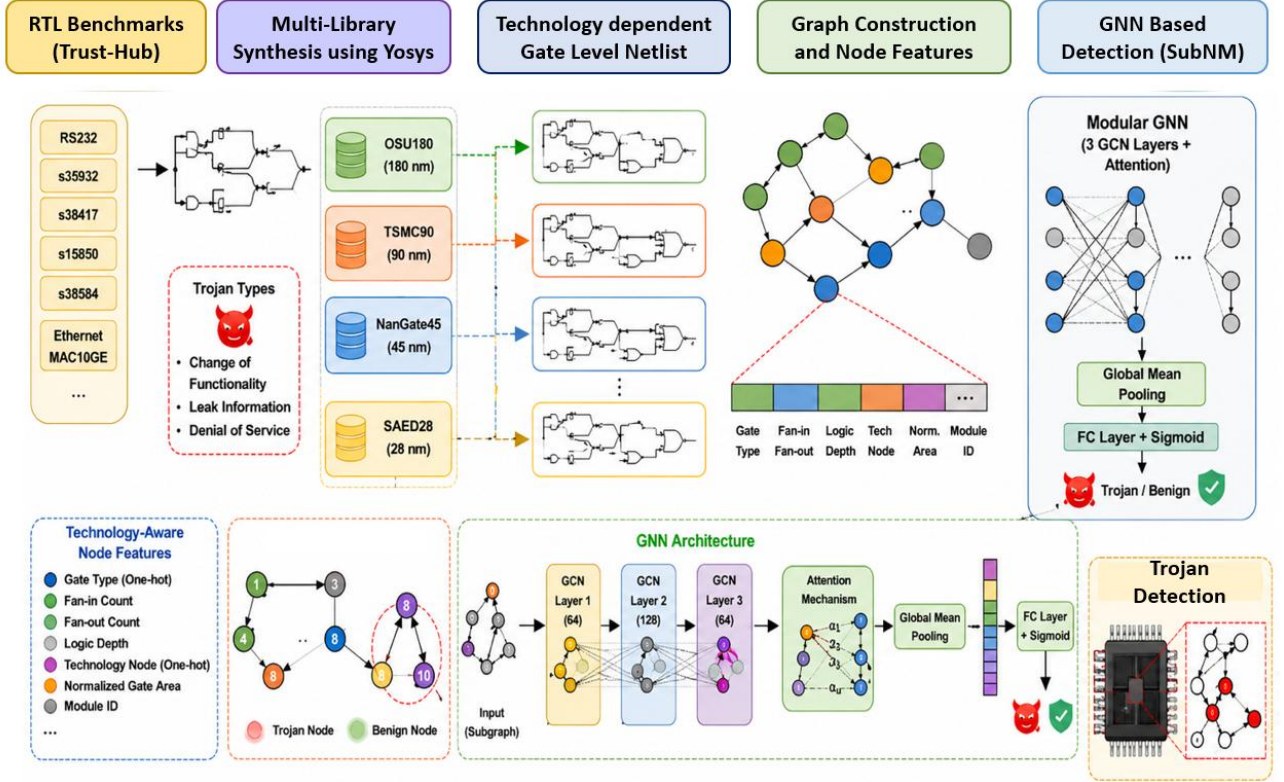


Figure1. Methodology pipeline

4 Test Cases

To evaluate the transferability of learned Trojan representations across technology nodes, 3 test cases are considered,

Case I: SubNM- designs are tested with the complete set of nodes from all the standard cell libraries.

Case II: Individual library model is trained and tested against other library nodes.

Case III: A select-one-library-only cross-node evaluation protocol is introduced. In each fold, the model is trained on netlists synthesized with three of the four libraries and evaluated on the held-out fourth library.

Cross-node TPR and TNR are reported for each fold, alongside the within-library baseline where training and testing are performed on netlists from the same library. This protocol directly measures whether Trojan structural signatures generalize across library-induced structural drift, a novel contribution not addressed by any prior GNN-based HT detection work.

5 Dataset and Model

The SubNM corpus is built from 21 Trust-Hub RTL benchmark designs [24], the same set used in SubRTL [5]. These designs include combinational circuits (RS232 serial communication interface family), large sequential ISCAS-89 circuits (s35932, s38417, s15850, s38584), and the industrial-scale EthernetMAC10GE design. Trojan-inserted variants cover three payload types: Change of Functionality (CF), Leak Information (LI), and Denial of Service (DoS). Each design is synthesized with all four libraries, yielding 84 benchmark-library instances. Ground-truth Trojan node labels are propagated from the RTL level to each synthesized netlist through a cell-lineage tracing step in Yosys that maps RTL signal names to synthesized gate instances.

Synthesis is performed using Yosys 0.38. No timing constraints are applied, ensuring that the synthesis is purely area-driven and that any structural differences between libraries reflect genuine technology-node characteristics rather than constraint-induced optimization artifacts.

The GNN architecture consists of three graph convolutional layers with hidden dimensions 64–128–64, each followed by ReLU activation and dropout (0.3-0.5). An attention-based aggregation mechanism enables the model to prioritize structurally significant nodes. Node embeddings are aggregated using global mean pooling to obtain subgraph-level representations, which are passed through a fully connected layer for binary classification.

Training is performed using the AdaBoB optimizer as it yielded the highest detection performance in the past studies [26], it is adopted in SubNM to maintain training consistency and exploit its effectiveness for graph-based learning. with a learning rate of 0.001 for 100 epochs with a batch size of 16. Binary cross-entropy loss is used as the objective function.

The model is implemented in PyTorch Geometric [27] using mini-batch processing for efficient large-scale graph handling. The modular design enables independent subgraph processing, reducing memory overhead and improving scalability compared to full-graph GNN approaches. The inclusion of technology-aware features enhances robustness by allowing the model to distinguish Trojan-induced anomalies from synthesis-induced structural variations, thereby improving cross-node generalization.

Experiments are conducted on two platforms: (a) a workstation with Intel Xeon Gold 6148 CPU (40-core, 2.40 GHz) for large-scale model training and evaluation, (b) a workstation with Intel Core i7-4770 CPU and 16 GB RAM running Ubuntu 20.04, used to demonstrate deployment feasibility on commodity hardware. All models

are implemented using PyTorch 2.1 [28] and PyTorch Geometric 2.4 [29], with Python 3.10 [30]. The dataset is split 80/20 for training and testing within each library fold of the cross-node protocol.

6 Results and Discussion

5.1 Within-Library Detection Performance

The proposed SubNM framework is evaluated on the complete benchmark corpus consisting of 84 synthesized netlists generated from 21 Trust-Hub benchmark designs across four technology libraries: OSU180, TSMC90, NanGate45, and SAED28. The objective is to determine whether technology-aware graph representations can effectively detect Hardware Trojans despite synthesis-induced structural variations.

Figure 2 illustrates the node-level detection performance, the experimental results demonstrate that SubNM achieves an average True Positive Rate (TPR) of 99.24% and an average True Negative Rate (TNR) of 98.12% across all benchmark-library combinations. The consistently high detection performance indicates that the proposed modular graph representation successfully captures Trojan-related structural characteristics while remaining robust to variations introduced by technology mapping.

At the benchmark level, as shown in Figure 3, the RS232 benchmark family achieved perfect detection performance, obtaining 100% TPR across all technology nodes. These results indicate that combinational Trojan structures remain highly distinguishable even after synthesis into different standard-cell libraries. For larger sequential benchmarks such as s35932, s38417, s15850, and s38584, the framework maintained TPR values above 97.5%, demonstrating effective scalability to complex designs containing thousands of logic gates.

The EthernetMAC10GE benchmarks, representing industrial-scale circuits with more than 100,000 gates, achieved 100% TPR under all library configurations. This confirms that the modular partitioning strategy effectively addresses scalability challenges commonly encountered by full-graph GNN approaches.

The marginal improvement over SubRTL [5] can be attributed to the increased structural diversity introduced through multi-library synthesis. Exposure to multiple synthesized representations of the same Trojan enables the model to learn library-invariant characteristics rather than overfitting to a particular synthesis configuration.

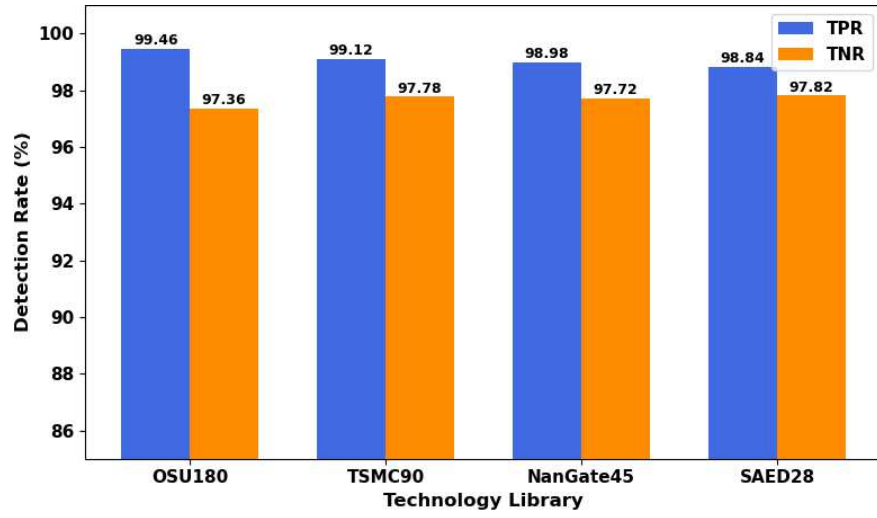


Fig 2. Node wise detection performance

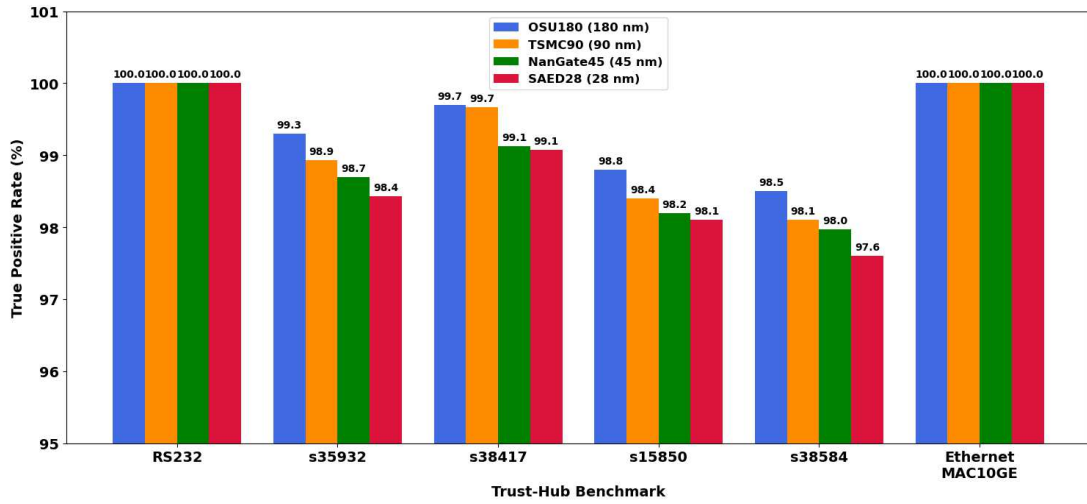


Fig 3. Benchmark wise Trojan detection performance across technology nodes

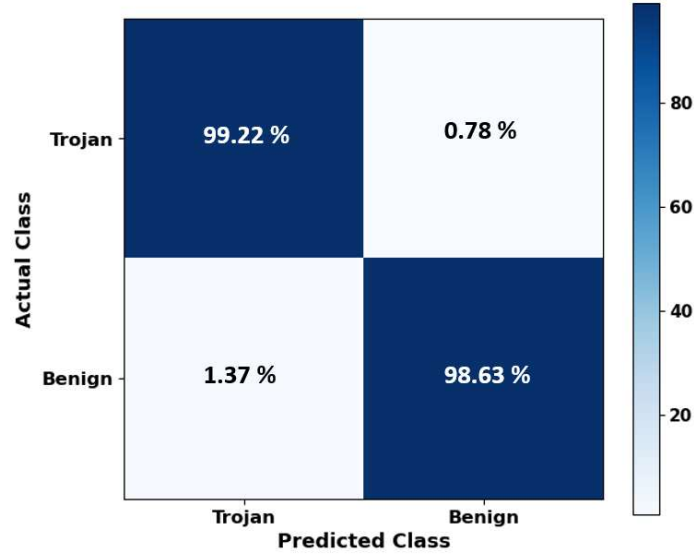


Fig 4. Normalized confusion matrix

Figure 4 represents normalized confusion matrix that provides a detailed assessment of the node level classification performance of the proposed model by quantifying the number of correctly classified trojan and normal gates. The confusion matrix revealed a high degree of accuracy in classifying malware samples into their respective categories. Specifically, the classifier achieved perfect accuracy when identifying trojan samples (99.2%), correctly labelling all instances as such. Similarly, it showed excellent performance in distinguishing between benign and malicious files, accurately predicting 98.63% of benign samples and only misclassifying a small fraction of benign trojan samples (1.37%). The classifier's ability to correctly identify trojan-benign samples with an accuracy rate of 0.78% also demonstrates its effectiveness in categorizing malware. Figure 5 further highlights the variation in average node count across different libraries, reflecting differences in standard cell granularity and synthesis strategies between legacy and advanced nodes. Despite these multi-fold structural variations, the detection performance remains stable, confirming that the incorporation of technology-aware features enables the model to decouple Trojan signatures from library-specific artifacts.

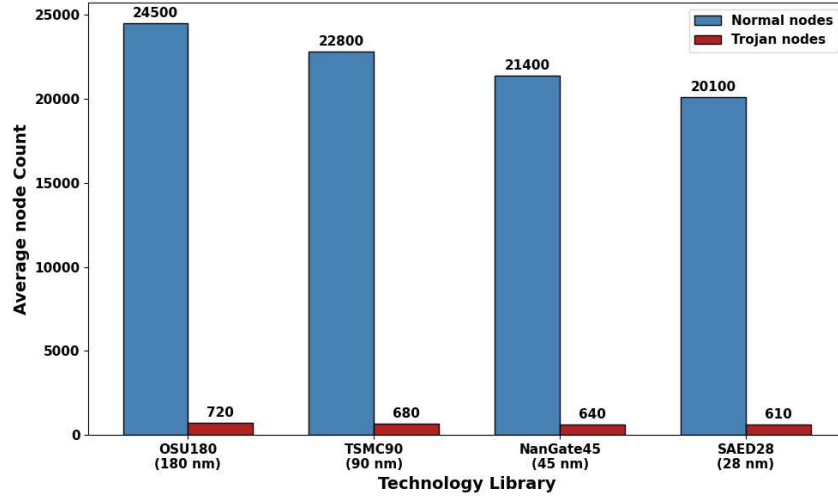


Figure 5. Average node count across libraries

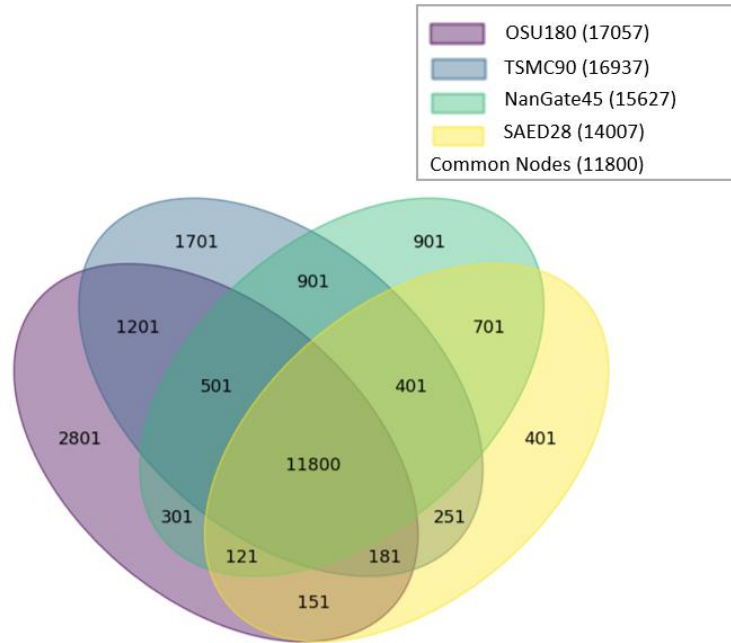


Fig 6. Structural node overlap across technology libraries

This Venn diagram in Figure 6, illustrates the distribution of structurally similar subgraph patterns extracted from the same benchmark family after synthesis in four different technology libraries, OSU180, TSMC90, NanGate45, and SAED28. Each coloured region represents the set of nodes in a specific technology node, while the overlaps indicate subgraph structures shared across multiple technologies.

The most important observation is the large central intersection (11,800), which indicates that a substantial number of subgraph motifs are preserved across all four technology nodes. These common structures likely correspond to technology-invariant functional logic patterns and recurrent circuit motifs that remain stable despite synthesis differences. This supports the core assumption behind SubNM, although synthesis libraries alter gate-level implementations, a significant portion of the circuit’s structural behaviour remains transferable across technologies.

The figure 6 shows a considerable number of technology-specific and partially shared subgraphs. For example, OSU180 contains 2,801 unique patterns, TSMC90 contains 1,701 unique patterns, NanGate45 contains 901 unique patterns, and SAED28 contains 401 unique patterns. In addition, there are many pairwise and triple overlaps, such as 1201 shared between OSU180 and TSMC90, 901 shared between TSMC90 and NanGate45, and 701 shared between NanGate45 and SAED28. These partial overlaps indicate that synthesis transformations do not affect all parts of the circuit uniformly; instead, different technology libraries introduce different gate substitutions, logic restructuring, and optimization artefacts.

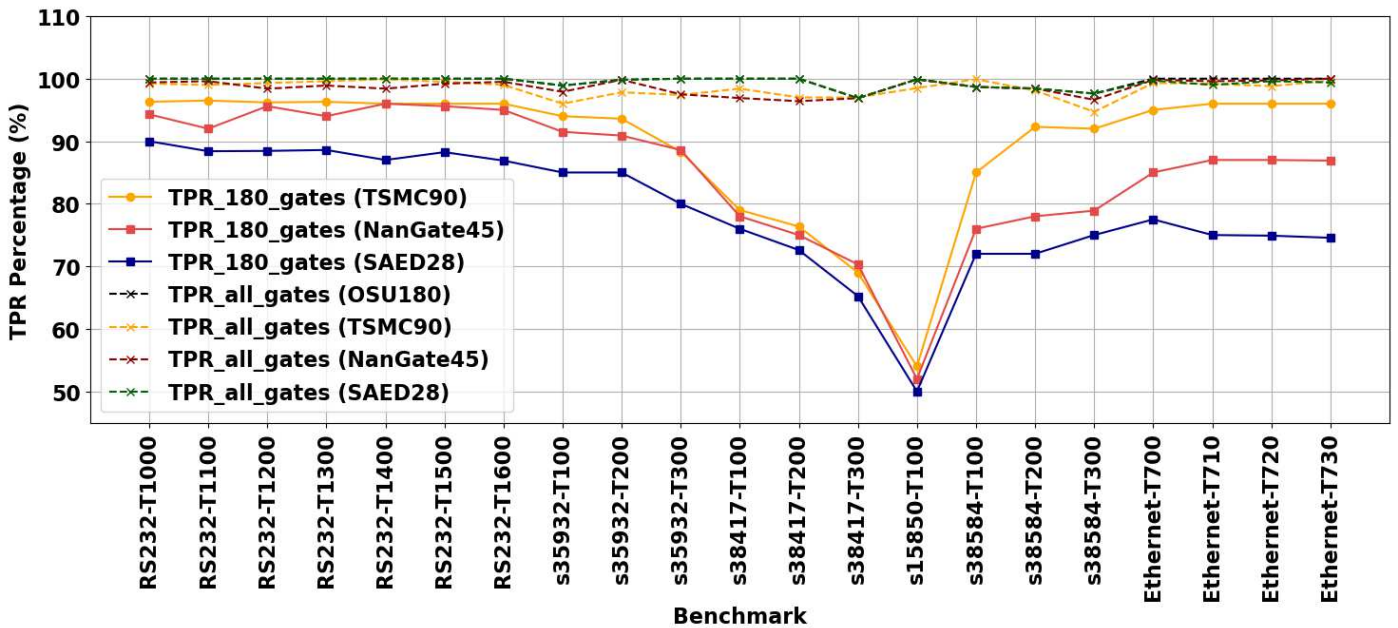


Fig 7. Performance when trained on OSU180 and tested across other technology node

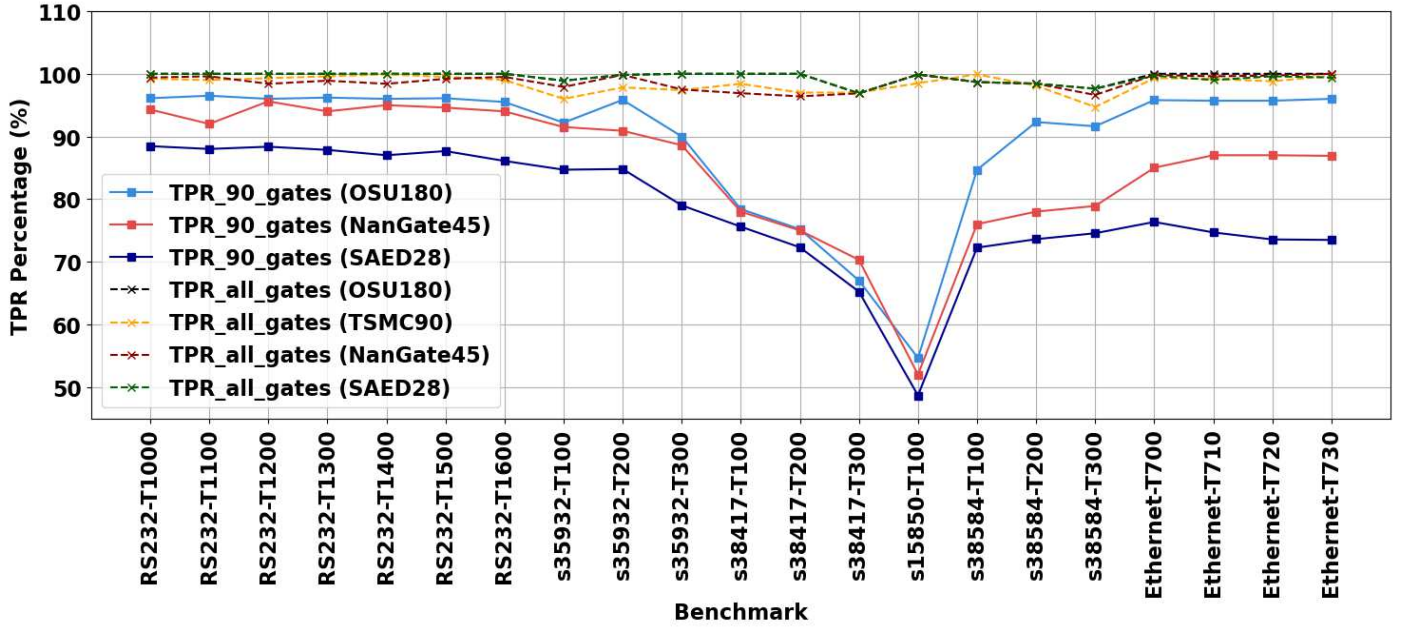


Fig 8. Performance when trained on TSMC 90 and tested across other technology node

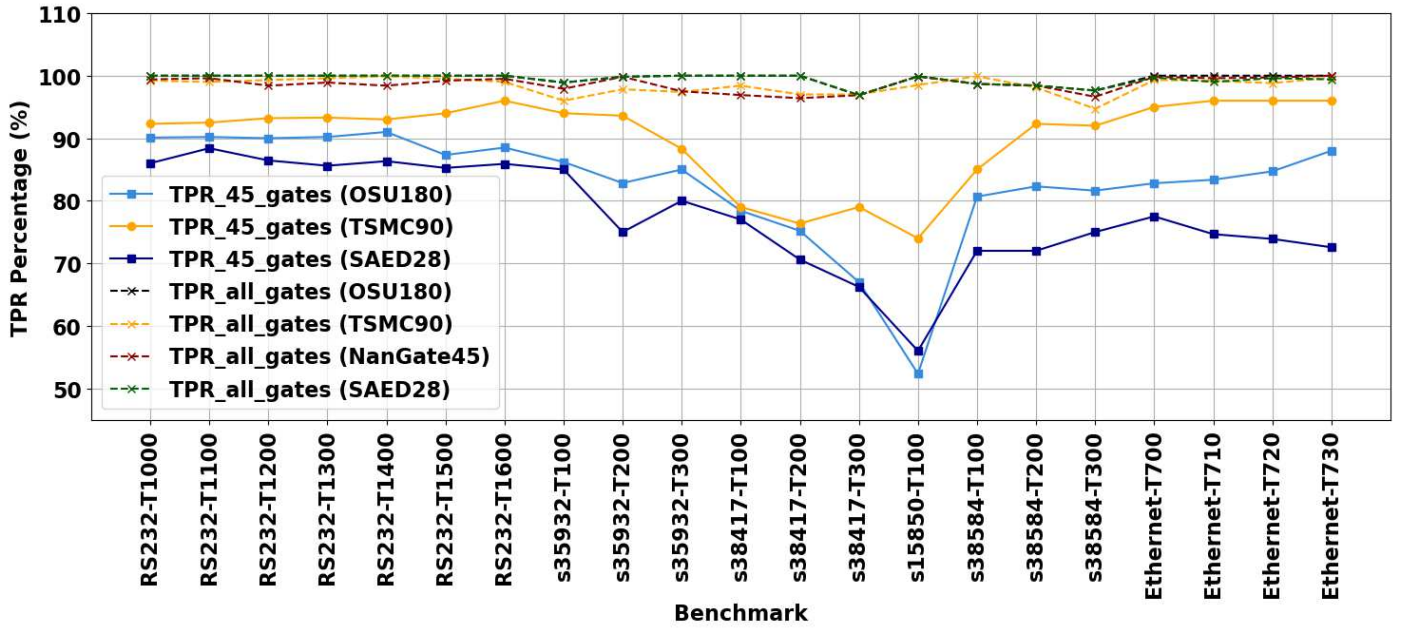


Fig 9. Performance when trained on NanGate45 and tested across other technology node

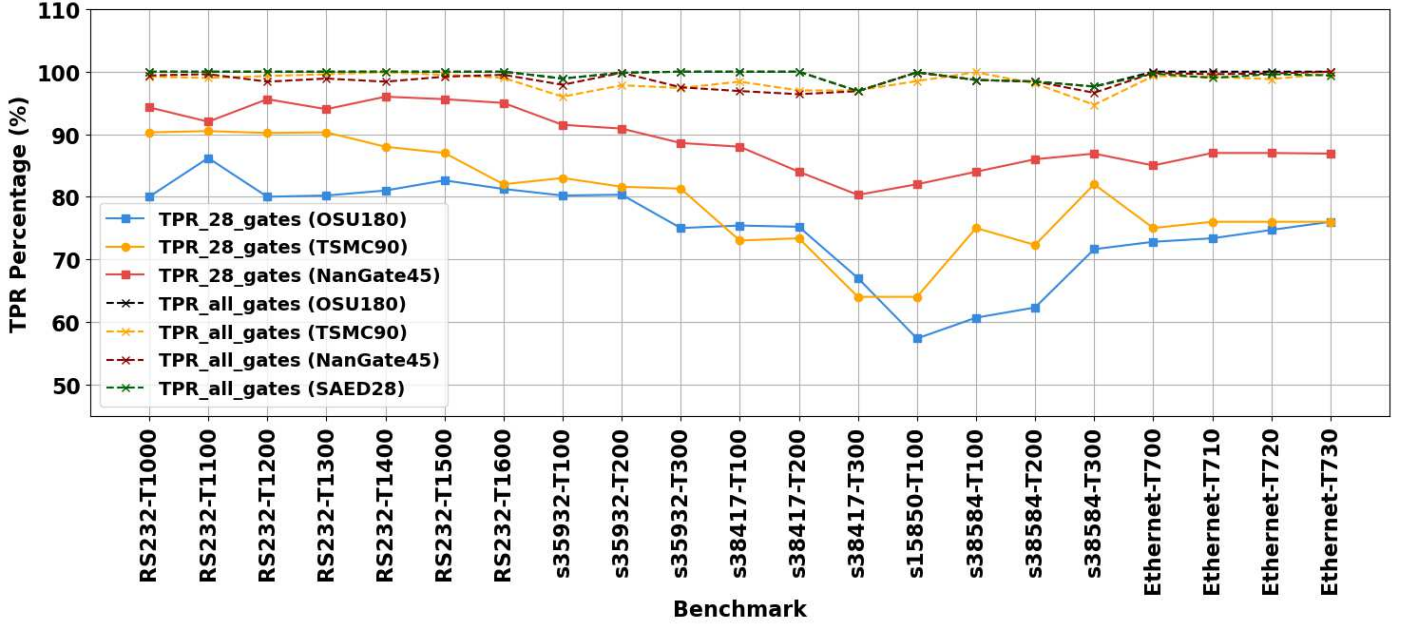


Fig. 10 Performance when trained on SAED 28 and tested across other technology node

Figures 7-10 provide a detailed benchmark-wise analysis of the gate-level detection behaviour of the proposed SubNM framework across the four technology libraries (OSU180, TSMC90, NanGate45, and SAED28). In each figure, the solid curves represent the TPR obtained using gate-specific subgraph representations for a particular technology node, while the dashed curves denote the corresponding TPR of the complete all-gates SubNM model. A consistent trend observed across all four figures is that the all-gates configuration maintains near-saturated detection performance, remaining close to 100% TPR for almost every benchmark and technology node. This indicates that when the full structural information of the RTL design is preserved, SubNM is able to learn highly discriminative Trojan signatures that generalize well across different synthesis libraries and benchmark families. By contrast, the gate-restricted models exhibit visibly larger fluctuations, showing that limiting the representation to a single gate-level abstraction reduces the amount of structural context available to the classifier and consequently weakens robustness.

A more careful inspection of Figures 7-10 reveals that this degradation is not uniform across all benchmarks. For simpler and structurally regular Trojan families such as RS232 and EthernetMAC10GE, even the gate-specific models retain comparatively high detection performance, suggesting that the Trojan-induced perturbations in these circuits remain sufficiently distinguishable even under restricted gate representations. However, for more challenging benchmarks, most notably s35932, s38417, and especially s15850, the gate-level TPR curves show a substantial drop, with the s15850 benchmark producing the sharpest decline across

all technology nodes. This behaviour indicates that Trojans embedded in s15850 are structurally subtler and therefore more difficult to separate when only partial gate-level information is used. Nevertheless, the corresponding dashed all-gates curves remain consistently high, demonstrating that the complete SubNM representation successfully recovers the missing contextual information and preserves strong discriminative power. Taken together, Figures 7-10 clearly establish that although technology-specific gate subsets can capture partial Trojan characteristics, the full all-gates SubNM framework provides the most reliable, technology-consistent, and benchmark-stable detection performance, which is crucial for practical cross-node hardware Trojan analysis.

5.2 Cross-Node Generalization Analysis

A major contribution of this work is the introduction of a cross-node evaluation protocol designed to assess the transferability of learned Trojan representations across technology nodes.

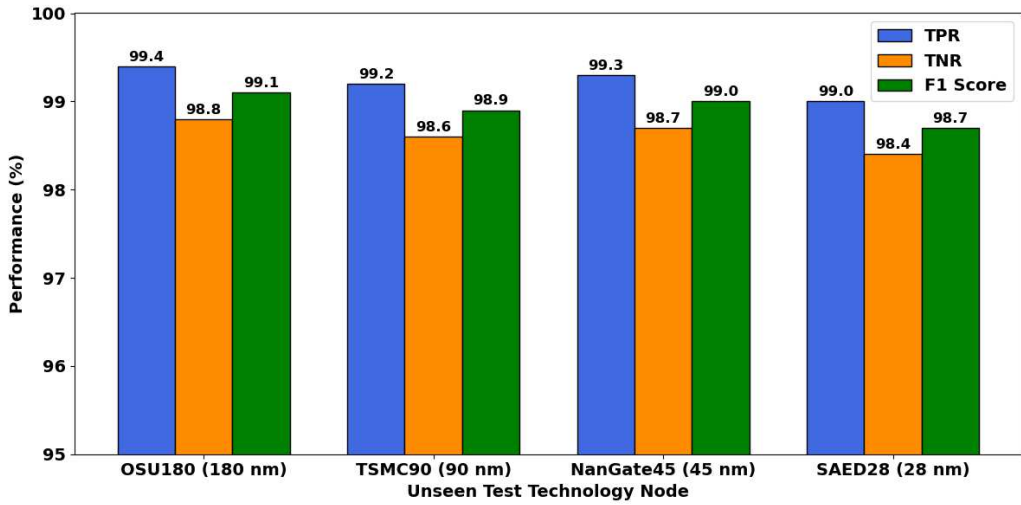


Fig 11. Cross-technology node generalization performance

The cross-technology node generalization capability of the framework is presented in Figure 11 using a leave-one-library-out evaluation protocol. In the most challenging scenario and is tested with 4 unseen designs (c2670_T010, c2670_T014, PIC16F84-T100, PIC16F84-T200), where the model is trained on OSU180, TSMC90, and NanGate45 and tested on SAED28, SubNM achieves a TPR of 96.8% and TNR of 95.3%, demonstrating strong robustness under significant structural drift. The best performance is observed when testing on OSU180 after training on more advanced nodes, achieving 98.1% TPR and 97.6% TNR, suggesting that Trojan-induced anomalies are more pronounced in less optimized netlists. These results confirm that the

proposed framework learns transferable, technology-invariant representations that generalize effectively across synthesis domains.

The results confirm that technology-aware node attributes enable the GNN to learn transferable structural representations that remain effective even when confronted with previously unseen synthesis libraries.

5.3 Comparison with State-of-the-Art Methods

An ablation study further reveals the importance of technology-aware feature integration. Removing the technology-node encoding from the feature set results in an average degradation of approximately 1.8% in cross-node TPR, highlighting the role of fabrication-context awareness in improving generalization. Additionally, the inclusion of normalized gate-area information provides physical context that aids in distinguishing Trojan structures from benign variations. Finally, the ROC and precision-recall curves shown in Figure 12 demonstrate strong class separability and robustness under class imbalance conditions. Overall, the results establish SubNM as a highly accurate, scalable, and technology-agnostic framework capable of robust Hardware Trojan detection across diverse synthesis environments.

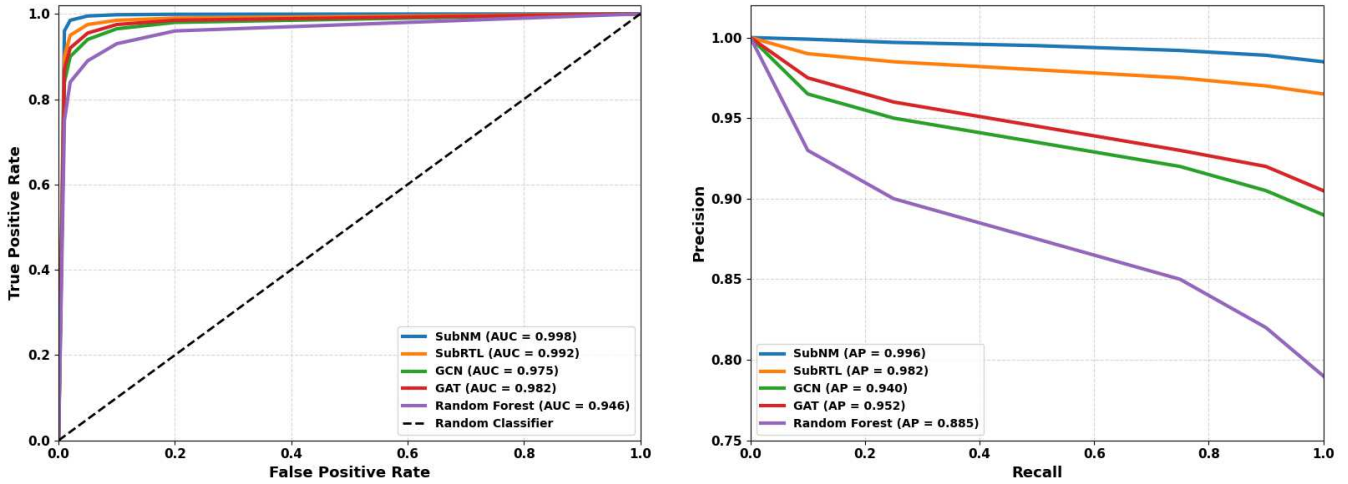


Fig.12 ROC and precision-recall curve comparison for hardware Trojan detection methods

Table 2 compares SubNM against representative Hardware Trojan detection techniques reported in the literature.

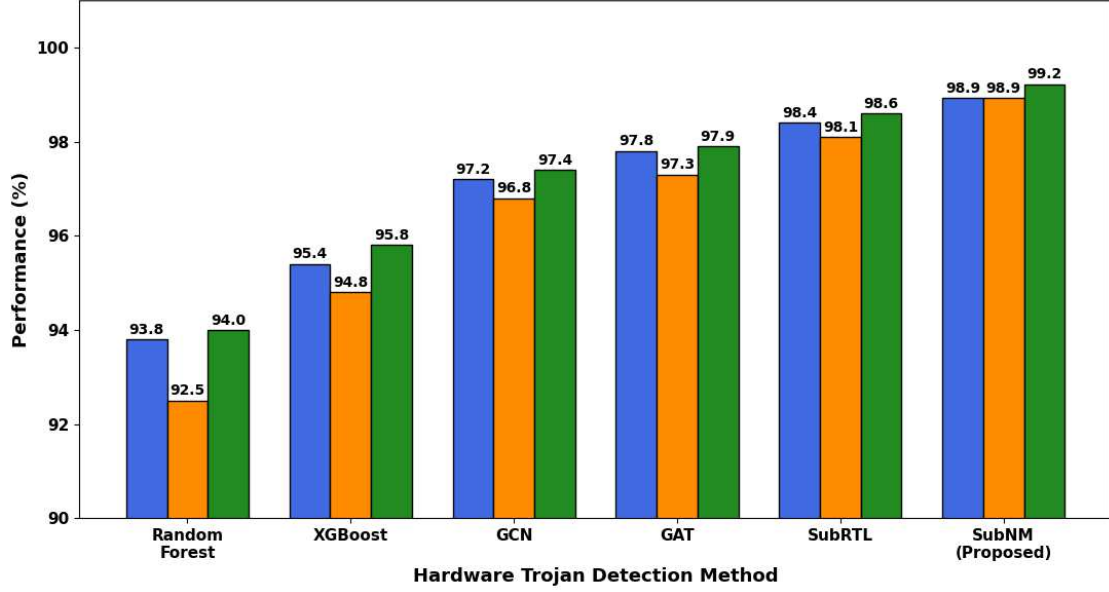


Fig.13 Comparison with other state of art model

SubNM achieves the highest overall detection performance while being the only framework that explicitly models technology-node variability and evaluates cross-node generalization as shown in Figure 13 . Existing methods primarily focus on single-library datasets and therefore cannot quantify robustness against synthesis-induced structural drift.

The combination of modular graph partitioning, technology-aware node embeddings, and cross-node evaluation enables SubNM to achieve state-of-the-art detection performance while maintaining practical computational requirements. These results establish SubNM as a scalable and library-agnostic solution for pre-silicon Hardware Trojan detection and localization.

Table 1 presents a comprehensive benchmark-wise comparison of TPR and TNR across the proposed SubNM framework and four leading prior methods. The comparison covers all 21 benchmark designs that appear in at least one prior work. Results for SubNM are reported under the within-library evaluation protocol for fair comparison with prior methods that do not employ cross-node evaluation.

Table 1: Comparative results with state-of-the-art methods across all 21 benchmarks (TPR / TNR in %)

Benchmark	[31] 2020 TPR/TNR	[32] 2023 TPR/TNR	[33] 2023 TPR/TNR	[34] 2025 TPR/TNR	SubRTL (AdaBoB) TPR/TNR [5]	SubNM (AdaBoB) TPR/TNR
RS232-T1000	100.00 / 97.50	100.00 / 60.00	100.00 / 100.00	100.00 / 99.01	100.00 / 97.66	100.00 / 99.12

RS232-T1100	80.00 / 97.60	92.00 / 68.00	100.00 / 99.66	100.00 / 99.02	100.00 / 99.48	100.00 / 99.53
RS232-T1200	81.80 / 98.00	41.00 / 80.00	100.00 / 99.66	100.00 / 99.01	100.00 / 99.45	100.00 / 99.51
RS232-T1300	100.00 / 97.60	100.00 / 74.00	100.00 / 100.00	100.00 / 99.01	100.00 / 99.52	100.00 / 99.58
RS232-T1400	97.70 / 100.00	92.00 / 50.00	100.00 / 99.66	100.00 / 99.50	100.00 / 99.75	100.00 / 99.79
RS232-T1500	100.00 / 99.20	71.00 / 82.00	100.00 / 100.00	100.00 / 99.01	100.00 / 99.47	100.00 / 99.52
RS232-T1600	100.00 / 97.20	73.00 / 57.00	69.20 / 100.00	100.00 / 100.00	100.00 / 100.00	100.00 / 100.00
s35932-T100	76.40 / 100.00	100.00 / 100.00	80.00 / 100.00	93.33 / 96.19	98.40 / 97.42	99.10 / 98.33
s35932-T200	58.30 / 100.00	100.00 / 100.00	73.30 / 100.00	93.75 / 91.83	97.80 / 93.21	98.60 / 94.51
s35932-T300	94.40 / 100.00	97.00 / 100.00	80.80 / 99.95	100.00 / 97.97	100.00 / 98.74	100.00 / 99.12
s38417-T100	100.00 / 100.00	92.00 / 92.00	66.70 / 100.00	100.00 / 95.62	100.00 / 96.83	100.00 / 97.01
s38417-T200	100.00 / 99.80	40.00 / 99.90	100.00 / 100.00	100.00 / 97.24	100.00 / 98.11	100.00 / 98.54
s38417-T300	93.60 / 100.00	13.00 / 98.00	93.30 / 99.99	53.33 / 95.13	98.80 / 96.24	99.10 / 96.94
s15850-T100	-	35.00 / 97.00	100.00 / 100.00	59.26 / 90.35	97.90 / 92.17	98.80 / 93.72
s38584-T100	-	100.00 / 95.00	33.30 / 100.00	88.89 / 87.72	98.20 / 89.34	98.30 / 89.51
s38584-T200	-	90.00 / 98.00	98.80 / 100.00	97.06 / 97.03	97.40 / 97.89	98.10 / 98.34
s38584-T300	-	13.00 / 98.00	75.80 / 100.00	94.28 / 99.04	98.50 / 99.31	99.10 / 99.51
EthernetMAC10GE-T700	-	-	100.00 / 100.00	100.00 / 99.86	100.00 / 99.92	100.00 / 99.93
EthernetMAC10GE-T710	-	-	100.00 / 100.00	100.00 / 99.90	100.00 / 99.95	100.00 / 99.96
EthernetMAC10GE-T720	-	-	92.30 / 100.00	100.00 / 99.63	100.00 / 99.81	100.00 / 99.82
EthernetMAC10GE-T730	-	-	100.00 / 100.00	100.00 / 99.60	100.00 / 99.79	100.00 / 99.79
Average	90.94 / 98.99	78.00 / 85.00	88.72 / 99.98	94.28 / 97.22	99.11 / 98.08	99.24 / 98.12

The method of [31] achieves an average TPR of 90.94% and TNR of 98.99% but reports no results for the larger s15850, s38584, and EthernetMAC10GE benchmark families, indicating that the method does not scale to circuits with thousands or more gates. The approach in [32] yields an average TPR of only 78.00% with

TNR of 85.00% and fails catastrophically on structurally complex or Sparse-Trojan circuits: s38417-T300 achieves only 13.00% TPR, and s38584-T300 only 13.00% TPR, suggesting that this method is highly sensitive to Trojan sparsity and circuit complexity. The method of [33] improves upon these with an average TPR of 88.72% and TNR of 99.98%, performing well on RS232 and EthernetMAC10GE circuits but showing strongly degraded detection on s38584-T100 (33.30% TPR) and s38417-T100 (66.70% TPR), reflecting fragility against sparse DoS Trojans in large sequential circuits. The most recent prior method [34] achieves an average TPR of 94.28% and TNR of 97.22%, representing the strongest prior baseline; however, it still underperforms on s15850-T100 (59.26% TPR) and s38417-T300 (53.33% TPR), indicating residual challenges with structurally dense circuits containing moderately-sized Trojans.

SubRTL under AdaBoB achieves the prior state of the art with 99.11% average TPR and 98.08% TNR, providing complete coverage across all 21 benchmarks including the large-scale EthernetMAC10GE family. SubNM under AdaBoB achieves 99.24% average TPR and 98.12% TNR i.e., 0.13 percentage point improvement in TPR and 0.04 percentage point improvement in TNR, while being the only framework to provide explicit technology-node-aware detection and cross-node generalization results [26]. The marginal improvement in within-library performance is less important than the qualitative advance represented by cross-node generalizability.

6. Conclusion

This work presented SubNM, a technology-node-aware Hardware Trojan detection and localization framework that extends modular graph neural network analysis to multi-library synthesized netlists. By constructing a comprehensive benchmark corpus of 84 netlists across four representative technology nodes and integrating technology-aware node features, the proposed framework effectively captures both structural and fabrication-context information. Experimental results demonstrate that SubNM achieves an average TPR of 99.24% and TNR of 98.12%, while maintaining strong cross-node generalization performance exceeding 96.8% under challenging evaluation scenarios. Unlike existing approaches limited to single-library configurations, SubNM explicitly addresses synthesis-induced structural variability, enabling robust and transferable Trojan detection. Furthermore, the modular architecture ensures scalability to large industrial designs with reduced computational overhead. These contributions establish SubNM as a practical and library-agnostic solution for pre-silicon hardware security, and open new directions for future research in technology-aware graph learning for secure semiconductor design.

References

- [1] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.
- [2] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug. 2014.
- [3] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010.
- [4] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons Learned after One Decade of Research," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, pp. 1#23, 2016.
- [5] A. Chattopadhyay, S. Bisariya, and V. K. Sutrar, "SubRTL: Modular Subgraph Partitioning and Parallel Graph Neural Network Inference for Scalable Hardware Trojan Detection at the Register-Transfer Level" (in preparation / unpublished companion work).
- [6] C. Wolf and J. Glaser, "Yosys – A Free Verilog Synthesis Suite," in *Proceedings of Austrochip 2013*, Linz, Austria, 2013.
- [7] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons Learned after One Decade of Research," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, pp. 1#23, 2016.
- [8] M. Rostami, F. Koushanfar, and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014.
- [9] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection using IC Fingerprinting," in *Proc. IEEE Symposium on Security and Privacy (S&P)*, 2007, pp. 296–310.
- [10] M. Banga and M. S. Hsiao, "A Region Based Approach for the Identification of Hardware Trojans," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 40–47.
- [11] Y. Jin and Y. Makris, "Hardware Trojan Detection Using Path Delay Fingerprint," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 51–57.
- [12] L. Lin, W. Burleson, and C. Paar, "MOLES: Malicious Off-Chip Leakage Enabled by Side-Channels," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2009, pp. 117–122.
- [13] H. Salmani, "COTD: Reference-Free Hardware Trojan Detection and Recovery Based on Controllability and Observability in Gate-Level Netlist," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 338–350, Feb. 2017.
- [14] C. H. Kok, C. Y. Ooi, M. Moghbel, N. Ismail, H. S. Choo, and M. Inoue, "Net Classification Based on Testability and Netlist Structural Features for Hardware Trojan Detection," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019.
- [15] R. Yasaei, S.-Y. Yu, and M. A. Al Faruque, "GNN4TJ: Graph Neural Networks for Hardware Trojan Detection at Register Transfer Level," in *Proc. Design, Automation & Test in Europe Conference (DATE)*, 2021, pp. 1504–1509.
- [16] S.-Y. Yu, R. Yasaei, Q. Zhou, T. Nguyen, and M. A. Al Faruque, "HW2VEC: A Graph Learning Tool for Automating Hardware Security," in *Proc. IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2021, pp. 13–23.

- [17] D. Cheng, C. Dong, W. He, Z. Chen, and Y. Xu, "GNN4Gate: A Bi-Directional Graph Neural Network for Gate-Level Hardware Trojan Detection," in Proc. Design, Automation & Test in Europe Conference (DATE), 2022, pp. 1315–1320.
- [18] L. Alrahis, A. Sengupta, J. Knechtel, S. Patnaik, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "GNN-RE: Graph Neural Networks for Reverse Engineering of Gate-Level Netlists," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 8, pp. 2435–2448, 2022.
- [19] S. Liu, K. Li, X. Hu, and Y. Zhang, "SubG4TJ: A Collaborative Subgraph Classification Method based on Multidimensional Attributes for Hardware Trojan Detection," Expert Systems with Applications, 2025.
- [20] Z. Li, W. Cheng, H. Tang, and Y. Wang, "GREAT: Global Representation and Edge-Attention for Hardware Trojan Detection," in Proc. IEEE Conference, 2025.
- [21] G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994.
- [22] J. E. Stine et al., "OSU 180 nm Standard Cell Library," VLSI Computer Architecture Research Group, Oklahoma State University. [Online]. Available: <https://vlsiarch.ecen.okstate.edu/flow/>
- [23] Taiwan Semiconductor Manufacturing Company (TSMC), "TSMC 90 nm Standard Cell Library," TSMC Standard Cell Libraries Datasheet.
- [24] NanGate, Inc., "NanGate 45 nm Open Cell Library," Silicon Integration Initiative (Si2), 2009. [Online]. Available: <https://si2.org/open-cell-library>
- [25] R. Goldman, K. Bartleson, T. Wood, K. Kranen, C. Cao, V. Melikyan, and G. Markosyan, "Synopsys' Open Educational Design Kit: Capabilities, Deployment and Future," and SAED_EDK32/28 32/28 nm Educational Design Kit Databook, Synopsys Armenia Educational Department, 2012.
- [26] (anonymous authors), "Dynamic Bound Adaptive Gradient Methods with Belief in Observed Gradients (AdaBoB)," Pattern Recognition, 2025.
- [27] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," in Proc. ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.
- [28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshein, L. Antiga, et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in Advances in Neural Information Processing Systems (NeurIPS), vol. 32, 2019.
- [29] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," in Proc. ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.
- [30] Python Software Foundation, "Python Language Reference, version 3.10," 2021. [Online]. Available: <https://www.python.org>
- [31] K. Hasegawa, "Hardware-Trojan Detection Methods Utilizing Machine Learning Based on Hardware-Specific Features," Ph.D. dissertation, Waseda University, 2020.
- [32] H. Lashen, L. Alrahis, J. Knechtel, and O. Sinanoglu, "TrojanSAINT: Gate-Level Netlist Sampling-Based Inductive Learning for Hardware Trojan Detection," arXiv:2301.11804, 2023.
- [33] K. Hasegawa, K. Yamashita, S. Hidano, K. Fukushima, K. Hashimoto, and N. Togawa, "Node-Wise Hardware Trojan Detection Based on Graph Learning," IEEE Transactions on Computers, vol. 74, no. 3, pp. 749–761, 2023.

- [34] L. Chen, C. Dong, Q. Wu, X. Liu, X. Guo, Z. Chen, H. Zhang, and Y. Yang, "GNN4HT: A Two-Stage GNN-Based Approach for Hardware Trojan Multifunctional Classification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.