Hardware Trojan Detection on Gate Level Netlist*

*2025 CAD Contest Problem A

En-Ling Hsiung
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
doo1222.tw@gmail.com

I. Proposed Methodology

The proposed methodology for hardware trojan detection is designed as a multi-stage pipeline. The core components include (1) feature extraction from the circuit dataset, (2) a dual-model classification system, and (3) a final post-processing and prediction stage.

A. Feature Extraction

In our methodology, we represent each circuit as a graph where each gate is considered a node. For the gate-level analysis (and as the foundation for aggregated circuit-level features), we characterize every node i with a 17-dimensional feature vector, $\mathbf{F}_{gate_i}$. This vector is designed to capture the node's functional, structural, and topological properties within the netlist.

The components of this 17-dimensional vector are defined as follows:

- Gate Type (10 dimensions): We use a one-hot encoding scheme to represent the logical function of the gate. This vector component identifies the gate's type, including NOT, BUF, AND, OR, XOR, NAND, NOR, XNOR, DFF, constant value.
- DFF Port Connectivity (5 dimensions): A set of five binary flags indicating whether the gate is directly connected to a specific port of a D-type flip-flop (DFF). These features are:
  - Connection to DFF 'clk' clock port.
  - Connection to DFF "sn" asynchronous set port.
  - Connection to DFF "rn" asynchronous reset port.
  - Connection to DFF "q" output port.
  - Connection to DFF "d" data port.

  Each flag is 1 if a connection exists, and 0 otherwise.
- Distance to Primary Input (1 dimension): A numerical value representing the topological distance from the current gate to the nearest Primary Input (PI) of the circuit.
- Distance to Primary Output (1 dimension): A numerical value representing the topological distance from the current gate to the nearest Primary Output (PO) of the circuit.

This 17-dimensional vector $\mathbf{F}_{gate_i}$ serves as the direct input for our gate-level classifier (Model 2) and is used in aggregate form for the circuit-level classifier (Model 1).

B. Model Selection

Given that circuits are naturally represented as graphs (where gates are nodes and wires are edges), we employ Graph Neural Networks (GNNs) to learn representations that capture the complex topological relationships. We utilize a dual-model system, both based on the GraphSAGE architecture, to perform classification at two different granularities: circuit-level and gate-level.

Both models share a similar core architecture: a multi-layer GNN with 'SAGEConv' layers, 'BatchNorm' for stabilization, a custom node-wise attention mechanism, and residual connections to improve training.

1) Model 1: Circuit-Level Classifier: The objective of this model is to classify the entire circuit graph as either "Trojaned" or "Trojan-Free".

The model processes the entire graph and uses 'SAGEConv' layers enhanced with our attention mechanism to learn node features. After the final GNN layer, a global mean pooling operation aggregates all node embeddings into a single graph-level feature vector. This vector, representing the entire circuit, is then passed through a final Multi-Layer Perceptron (MLP) head to produce a binary classification.

2) Model 2: Gate-Level Classifier: The second model is a node classification GNN designed to predict whether each individual gate is "Trojan" or "Benign".

It mirrors the circuit-level model's GNN architecture (SAGEConv, attention, and residuals) to compute a final feature embedding for every node. However, it omits the global pooling layer. Instead, a final MLP head is applied directly to each node's embedding independently. This produces a per-gate classification, effectively creating a "heat map" of potential trojan locations within the circuit.

C. Inference and Post-Processing

First, the 17-dimensional feature vectors are extracted for every gate in the circuit, as described in

1) Neighbor Voting (Gate-Level Refinement): The initial gate-level predictions are refined using a neighbor voting mechanism to reduce noise and isolated false positives. For each gate i, we examine the predictions of its immediate neighbors (nodes connected by an edge in the circuit graph).

If more than half of the neighbors' predictions contradict the prediction for gate i, the prediction P_{gate_i} is flipped (i.e., "Trojan" $\rightarrow$ "Benign" or "Benign" $\rightarrow$ "Trojan"). This process yields a refined list of trojan gates and their total count, N_{trojan_gates}.

2) Final Decision Logic: Finally, a rule-based module combines the circuit-level prediction $P_{circuit}$ and the refined gate-level count N_{trojan_gates} to make a final declaration.

Our system declares the circuit as "Trojan-Free" (Benign) if either of the following conditions is met:

1) The circuit-level classifier (Model 1) predicts that the circuit is "Trojan-Free".
2) The total number of trojan gates found after neighbor voting (N_{trojan_gates}) is less than 15.

Conversely, the circuit is declared "Trojaned" (Malicious) only if the circuit-level classifier predicts "Trojaned" and the number of detected trojan gates is 15 or more. In this case, the list of gates from the refined gate-level model is reported as the location of the hardware trojan.