SMART: Graph Learning-Boosted Subcircuit Matching for Large-Scale Analog Circuits

Jindong Tu, Yapeng Li, Pengjia Li, Peng Xu, Qianru Zhang, Sanping Wan, Yongsheng Sun, Bei Yu, *Senior Member, IEEE*, and Tinghuan Chen, *Member, IEEE*

***Abstract*—Subcircuit matching in a large-scale analog circuit is a fundamental problem in VLSI computer-aided design (CAD). Existing approaches suffer from a poor scalability issue for a large-scale analog circuit. In this article, we propose a graph learning-boosted subcircuit matching framework for large-scale analog circuits named SMART, consisting of two stages. In the first stage, we customize hypergraph neural networks to map circuit topology for embedding space. Then, coarse subcircuit recognition is directly performed in the embedding space by geometric relations between the query circuit and all candidate subcircuits within the target circuit. In the second stage, a radial matching method, including device attribute matching, connection relationship matching and uniqueness-based matching, is customized to perform fine matching and obtain matches between interconnections and devices in the query circuit and candidate subcircuits. Experimental results show our SMART can outperform state-of-the-art search-based method VF3 and learning-based method NeuroMatch, and achieve the fastest speed. Specifically, using our framework for subcircuit matching can achieve up to 135× speedup with slight accuracy loss, and up to 7× speedup while maintaining 100% accuracy.**

***Index Terms*—Analog Circuit, machine learning, matching, verification.**

I. Introduction

SUBCIRCUIT matching within large-scale analog circuits represents a critical challenge in VLSI computer-aided design (CAD). Central to this issue is the identification of isomorphisms between a query circuit and subcircuits of a target circuit, particularly focusing on matching interconnections and devices. This problem is not only integral to circuit analysis and synthesis [1], [2], [3], [4], but also plays a pivotal role in physical design [5], [6], [7], [8], [9] and verification processes [10]. For instance, in the circuit analysis, a related collection of interconnected primitive devices is identified as a single high-level module and then replaced with the corresponding module [1], [2]. In the circuit verification, interface circuits, which bridge voltage domains, are particularly vulnerable to electrostatic discharge (ESD) events. Accurately identifying such circuits within large-scale designs is vital for effective ESD circuit verification [10]. Additionally, in the physical design, specific analog circuit topologies are found to pose various geometrical matching constraints (*e.g.*, symmetry, regularity, common-centroid) for the performance specification and circuit robustness [7]. Traditionally, subcircuit matching is approached as a subgraph matching problem. This contrasts with subgraph isomorphism [11] in that subgraph matching specifically seeks to establish correspondences between interconnections and devices across the query and target circuits. These problems are known to be NP-complete [2], [12], presenting significant computational challenges in VLSI CAD.

Approaches to subcircuit matching or recognition in VLSI CAD can be broadly categorized into mathematical optimization, search-based methods, and machine learning techniques. Mathematical optimization approaches typically frame subcircuit matching as a binary programming problem. In this formulation, the variables represent potential matches between interconnections and devices across the target and query circuits, with the objective of minimizing the graph distance between a matched subcircuit and the query circuit. Despite their theoretical robustness, these methods often struggle with scalability, and even the application of relaxation and approximation techniques does not fully mitigate this issue [2], [18]. On the other hand, search-based approaches utilize algorithms, such as breadth-first search (BFS) or depth-first search (DFS) to systematically explore the circuit topology. They aim to identify isomorphic or desired subcircuits throughout the traversal process [1], [8], [19], [20]. However, the expansive search space associated with large-scale analog circuits introduces significant scalability challenges. While pruning methods are employed to manage this vast search space, they frequently result in a compromise on the accuracy of the matching results [19]. In particular, if the search-based approach is extended to fuzzy matching, some matching rules have to be defined by users [22]. In other words, different subcircuit identification needs the user to define specific matching rules, posing a challenge for adaptability.

The advent of machine learning has introduced innovative alternatives to traditional subcircuit recognition techniques.

Received 3 July 2024; revised 15 November 2024 and 31 December 2024; accepted 5 March 2025. Date of publication 11 March 2025; date of current version 22 September 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB4402900; in part by the National Natural Science Foundation of China under Grant 62304197; and in part by HiSilicon. This article was recommended by Associate Editor G. G. E. Gielen. (*Jindong Tu and Yapeng Li contributed equally to this work.*) (*Corresponding author: Tinghuan Chen.*)

Jindong Tu, Yapeng Li, Pengjia Li, and Tinghuan Chen are with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhan), Shenzhen 518172, Guangdong, China (e-mail: chentinghuan@cuhk.edu.cn).

Peng Xu and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR.

Qianru Zhang, Sanping Wan, and Yongsheng Sun are with the COT Department, HiSilicon, Shenzhen 518129, China.

Digital Object Identifier 10.1109/TCAD.2025.3549701

TABLE I
COMPARISON OF DIFFERENT GRAPH-BASED CIRCUIT ANALYSIS METHODS

	Methods					
	Mathematical Programming [2], [18]	Search-based [8], [1]–[20]	Machine learning-based			Our SMART
			Boundary Identification [4], [13]–[15]	Device Classification [6], [9], [16], [17]	NeuroMatch [21]	
Scalability	Poor	Poor	Good	Good	Good	Good
Adaptability	Good	Good	Poor	Poor	Good	Good
Matching or Recognition Type	Exact	Exact	Fuzzy	Fuzzy	Fuzzy	Exact
Output Matches?[1]	Yes	Yes	No	No	No	Yes

[1] Matches between the interconnections and devices of the query circuit and those of the candidate subcircuits.

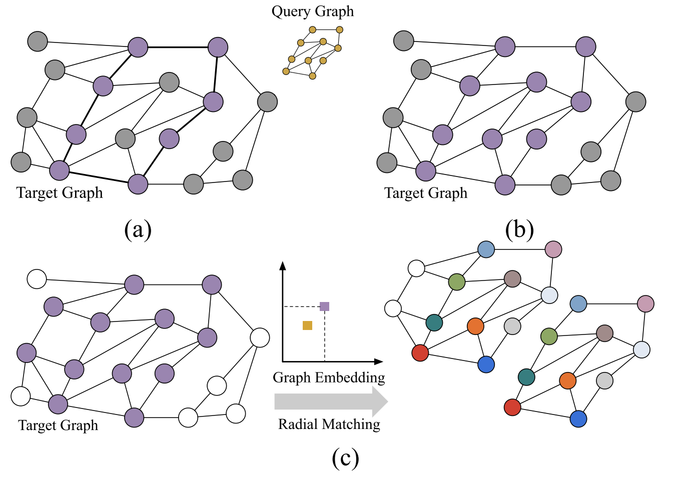


Fig. 1. Differences in subcircuit recognition and matching based on GNNs: (a) GNNs delineate the boundaries between different subcircuits within a circuit, with purple nodes indicating the identified subcircuit boundaries [4], [13], [14], [15]; (b) GNNs classify specific device groupings, where purple nodes represent devices within the identified subcircuits [6], [9], [16], [17]; (c) In our SMART, GNNs map the circuit topology into an embedding space. Subsequently, a RM method identifies matches between the query circuit and candidate subcircuits, with the graph formed by all purple nodes being mapped into the embedding space.

Given the natural representation of analog circuits as graphs, graph neural networks (GNNs) have emerged as a powerful tool for learning tasks on these structured data forms. In the VLSI CAD domain, GNNs have demonstrated considerable success, particularly when the graph structure and neural network architecture are meticulously designed to embody inductive biases tailored to specific VLSI CAD tasks [23], [24], [25]. GNNs facilitate subcircuit recognition through two primary methodologies. The first approach conceptualizes subcircuit recognition as a boundary identification problem, where GNNs are tasked with delineating the boundaries between different subcircuits within a circuit [4], [13], [14], [15], as shown in Fig. 1(a). The second approach views subcircuit recognition as a device classification, wherein the devices and interconnections are classified according to their membership in distinct analog subcircuits [6], [9], [16], or utilizes link prediction to classify specific device groupings [17], as shown in Fig. 1(b).

One of the limitations of using GNNs in this context is the necessity for retraining when new subcircuit types are introduced, posing a challenge for adaptability. While recent advancements, such as NeuroMatch have shown promise by efficiently recognizing subgraphs directly in the embedding space, thereby capturing geometric constraints relevant to subgraph relationships [21], machine learning approaches inherently struggle to achieve perfect recognition accuracy. Additionally, these machine learning methods do not typically provide direct matches between the interconnections and devices of the query circuit and those of the candidate subcircuits, hindering the analog circuit design automation applications. More specifically, in the circuit analysis, without matches between the query circuit and candidate subcircuits, a related collection of interconnected primitive devices cannot be automatically replaced by a single high-level module. In the physical design, without matches, geometrical constraints, *e.g.*, common centroid constraint [26], cannot be automatically posed. Consequently, while machine learning techniques like GNNs offer significant potential for improving subcircuit recognition, they currently do not fulfill all the criteria necessary for comprehensive and accurate subcircuit matching.

To tackle these challenges, this article introduces a novel graph learning-boosted framework for subcircuit matching in large-scale analog circuits, termed SMART. This work adopts a two-stage paradigm: first, leveraging GNNs to effectively capture circuit structural information, followed by search-based methods for exact matching. As depicted briefly in Fig. 1(c), SMART is designed to efficiently perform matching across various analog subcircuits without the need for retraining and user-defined rules. A comparison of different methodologies in subcircuit matching or recognition is summarized in Table I.

The framework is able to accept netlists of a large-scale target analog circuit and a query circuit as inputs and provides a comprehensive list of matched subcircuits, including detailed correspondences between interconnections and devices in both the query and target circuits. Given that analog circuits are often represented as hypergraphs, our approach employs hypergraph neural networks (HGNNs) as the core technology for subcircuit matching. Diverging from previous studies [1], [2], [4], [6], [8], [9], [13], [14], [15], [16], [17], [18], [19], [20], [22], the main idea of SMART is to employ HGNNs to quickly identify query-similar regions within the target graph, followed by exact subgraph matching through our search-based methods. Experimental evaluations demonstrate that SMART not only surpasses the state-of-the-art search-based and learning-based methods in terms of performance but also achieves an impressive speedup. This substantiates

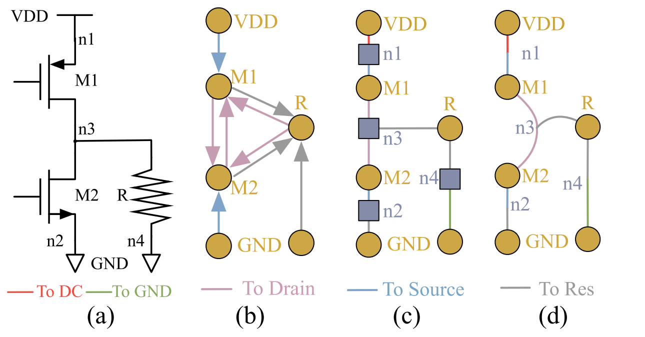


Fig. 2. Graph representation: (a) Analog circuit; (b) Directed multigraph; (c) Bipartite graph; (d) hypergraph.

the efficacy and efficiency of our proposed framework in addressing the complex challenges of subcircuit matching in large-scale analog circuits.

The article makes the following contributions.

1) For the first time, to the best of our knowledge, we present a graph learning-boosted framework that performs efficient subcircuit matching for large-scale analog circuits to obtain matches between interconnections and devices in a query circuit and target circuit.
2) We design an HGNN to map circuit topology for embedding space so that subcircuit matching is directly performed in the embedding space by training to capture geometric constraints corresponding to the relations between the query circuit and target subcircuits.
3) We develop a radial matching (RM) method to obtain matches between interconnections and devices in the query circuit and candidate subcircuits.
4) We conduct experiments on several large-scale analog circuits obtained by a topology synthesis tool, which confirms the accuracy and efficiency of our proposed framework compared with state-of-the-art search-based and learning-based methods.

The remainder of this article is organized as follows. In Section II, we give our problem formulation and preliminaries about the graph representation of analog circuits. We systematically present the proposed SMART framework in Section III, focusing on HGNNs and the RM algorithm. Section IV presents experimental results and discussion, followed by the conclusion in Section V.

II. Preliminaries

A. Graph Representation

As demonstrated in Fig. 2, an analog circuit can be naturally represented as a graph [23]. Traditional approaches convert an analog circuit into either a multigraph [4], [7], [17], [27], [28] or a bipartite graph [3], [5], [6], [9]. In the multigraph model, each device is depicted as a node, and interconnections are modeled as edges, with specific edge types corresponding to different port types, as illustrated in Fig. 2(b). The transformation from circuit to multigraph requires removing all nets while maintaining interconnections. Since this transformation is not one-to-one, it can lead to potential information loss and scalability issues. Bipartite graphs [3], [5], [6], [9] and hypergraphs [29], [30] address this limitation by explicitly modeling nets, either as nodes in bipartite graphs or as hyperedges in hypergraphs. A hyperedge is defined as an interconnection involving two or more nodes, where each hyperedge connects multiple devices in the circuit. While both representations effectively capture circuit topology and can be implemented using the same data structure, they lead to different message-passing mechanisms in neural network models. In bipartite graphs, messages must propagate through intermediate net nodes, introducing additional memory overhead and indirect message passing between devices. In contrast, hypergraphs and the corresponding HGNN model enable direct information exchange among devices connected by the same hyperedge.

Formally, our analog circuit hypergraph is defined as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ and $\mathcal{E}$ are the node set and hyperedge set, respectively. Each device is treated as a node $v \in \mathcal{V}$, and each net is treated as a hyperedge $e \in \mathcal{E}$. In particular, direct current (DC) source and ground (GND) nets typically connect to multiple devices across different circuit parts. Since GNN should only capture local structural features when converting subcircuit topology into embedding, treating DC and GND as regular nets would cause unwanted information aggregation between distant components during message passing, degrading matching performance. To address this issue, we segment DC and GND nets for each device and represent them as nodes, as shown in Fig. 2(a) and (d), where two GND nodes are connected to M2 and R through hyperedges n2 and n4, respectively. Additionally, since DC and GND types are essential characteristics for circuit matching, just like other device types, representing them as nodes allows us to naturally incorporate their information into the topology learning process. $\mathcal{R}$ represents the port type set since different port types must be distinguished for the subcircuit matching task. To distinguish interconnection, $\Psi = \{(v, e, r)\}$ represents the set of tuples and each element contains a node $v \in \mathcal{V}$, a hyperedge $e \in \mathcal{E}$ and their connection port type $r \in \mathcal{R}$. In practice, the connection port types include gate, drain, source, bulk, anode, cathode and others [30], [31]. The interconnection topology of the analog circuit hypergraph $\mathcal{G}$ can be represented by $|R|$ incidence matrices $\vec{H}_r$, where $|\cdot|$ is the set cardinality. When a node $\upsilon \in \mathcal{V}$ is connected by a hyperedge $e \in \mathcal{E}$ via the port type r, $h(v, e)_r = 1$, otherwise $h(v, e)_r = 0$.

B. Problem Formulation

In this article, we focus on addressing the subcircuit matching problem. We first give two definitions.

Definition 1 (Circuit Isomorphism): Two circuits are isomorphic if there exists a matching between their devices so that two devices are connected by an interconnection via device ports in one circuit if and only if corresponding devices are connected by an interconnection via the same device ports in the other circuit.

Definition 2 (Subcircuit): A subcircuit of a circuit is another circuit formed from a subset of the devices and their port interconnections.

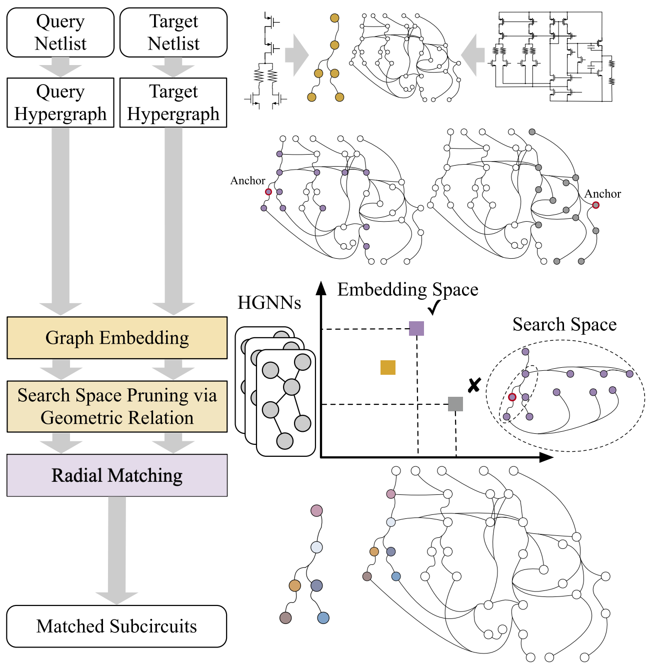


Fig. 3. Overall flow.

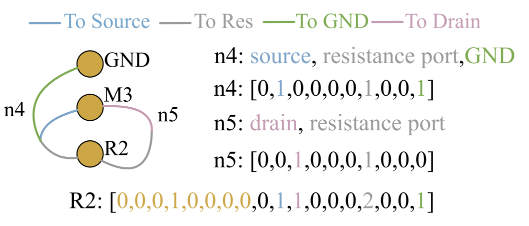

To Source
To Res
To GND
To Drain
GND
M3
R2
n4
n5
n4: source, resistance port,GND
n4: [0,1,0,0,0,0,1,0,0,1]
n5: drain, resistance port
n5: [0,0,1,0,0,0,1,0,0,0]
R2: [0,0,0,1,0,0,0,0,0,1,1,0,0,0,2,0,0,1]

Fig. 4. Node features.

Formally, we give our problem formulation as follows.

Problem 1 (Analog Subcircuit Matching): Given a query analog circuit and a large-scale target analog circuit, determine if the query analog circuit is isomorphic to subcircuits of the target analog circuit and obtain matches between interconnections and devices in the query analog circuit and all isomorphic subcircuits in the target analog circuit.

III. Proposed Method

A. Overall Flow

To handle Problem 1, we propose SMART, a graph learning-boosted subcircuit matching framework for large-scale analog circuits, as shown in Fig. 3. SMART takes netlists of a target circuit and a query circuit as inputs, and outputs all matched subcircuits with matches between nets and devices in the query circuit and target circuit. Initially, the target circuit and query circuit netlists are transformed into hypergraphs, respectively. For hierarchical circuit netlists, we will record hierarchical level information of devices during netlist parsing, thus we can locate identified subgraphs in the original structured netlist. To address the multiple-connection issue of DC and GND nets, during the transformation process, we partition these nets for each device. After partitioning, each DC or GND net would only connect to one device. Since a net must connect at least two devices, we represent DC and GND as nodes and add nets between these nodes and their previously connected devices. Our SMART consists of two main stages.

In the first stage, highlighted in yellow, we partition the target graph into several candidate subgraphs that potentially have isomorphic subgraphs to the query graph, then we employ HGNNs to map these subgraphs into an embedding space, where coarse subcircuit recognition is conducted directly in the embedding space. Here, the model is trained to identify geometric relationships between the query circuit and all candidate subcircuits within the target circuit. This strategy significantly enhances the efficiency and accuracy of the search space pruning process.

In the second stage, highlighted in purple, SMART implements a customized RM method, including device attribute matching, connection relationship matching and uniqueness-based matching. This method meticulously performs fine matching, establishing precise correspondences between the interconnections and devices in the query circuit and those in the candidate subcircuits.

B. Graph Embedding

Graph embedding encodes a graph into an embedding vector, allowing for coarse subgraph recognition by geometric relations. Our graph embedding approach includes preprocessing, information aggregation and the HGNNs model.

Preprocessing: During the preprocessing stage, the radius and center node of the query graph are determined by computing the maximum distance among all nodes. Subsequently, a BFS is conducted on the query graph, commencing from the center node, with the objective of enumerating the number of connected nets and identifying the type for each device encountered. In the target graph, a similar BFS procedure is employed to extract a subgraph of the same radius around all nodes possessing the identical type as the center node in the query graph. We can regard these nodes in the target as anchors. This extracted subgraph serves as the candidate subgraph and the aforementioned recording process is carried out in an analogous manner.

During the BFS process, a distinctive feature vector is assigned to each device (node), encompassing its device type and neighboring interconnections. Specifically, an 8-dimensional (8-D) one-hot vector is utilized to encode the device type, encompassing NMOS, PMOS, diode, resistance, capacitance, inductance, dc source, and GND. Concerning the neighboring interconnections, an initial 10-dimensional (10-D) vector is employed to encode the characteristics of each net connected to the device. Each element within this vector signifies the count of each port type connected, including gate, source, drain, bulk, anode, cathode, resistance port, capacitance port, dc source, and GND. Consequently, the final device feature vector is obtained by concatenating the 8-D one-hot representation with the summation of the 10-D vectors associated with all nets connected to the device. Fig. 4 shows the feature of the nets n4 and n5, and the device R2. In this way, the feature vector of the device includes its neighborhood information. Notably, the device and port types used in this feature vector construction are configurable, allowing users to define custom sets based on their specific analog circuit design requirements. We denote the feature vector of the ith device as $\vec{f}_i^{(0)}$. All device feature vectors in the analog circuit are stacked as a feature matrix $\vec{F}^{(0)}$.

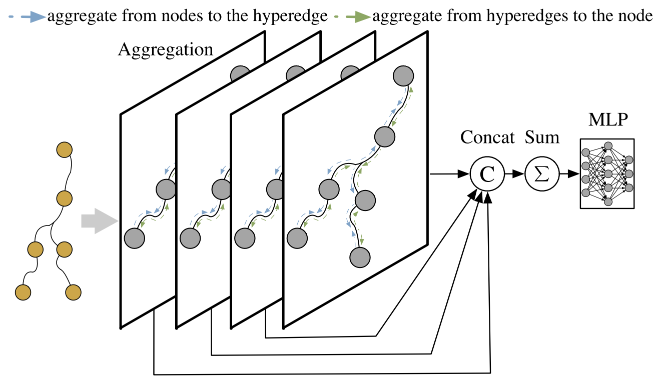
In the initial stage, candidate graphs are pruned based on prominent characteristics. For example, a candidate graph can be pruned if its number of nets or number of devices of any type is lower than that of the query graph. The initial pruning process leverages our defined device feature vector, as shown in Fig. 4. Leveraging the inherent characteristics of analog circuits, we capitalize on the fact that if the center device within the query circuit is successfully matched with the anchor in the candidate circuit, the number of connections of the same port type does not exceed that of its corresponding counterpart in the candidate circuit. Consequently, a subtraction operation is executed between the feature vectors of the two devices to validate this rule and carry out the initial candidate pruning.

Information Aggregation: We represent an analog circuit as a hypergraph, where each device is a node, and each net is a hyperedge [29], as shown in Fig. 2(d). To aggregate information from neighbors to the node itself in the hypergraph, we customize an aggregation operation

$$\vec{\boldsymbol{F}}_{\mathcal{N}}^{(l-1)} = \vec{\boldsymbol{D}}_e^{-1}\left(\sum_{r\in\mathcal{R}} w_r \vec{H}_r\right)\vec{\boldsymbol{D}}_v^{-1}\left(\sum_{r\in\mathcal{R}} \vec{\boldsymbol{H}}_r\right)^{\top}\vec{\boldsymbol{F}}^{(l-1)} \quad (1)$$

$$\vec{\boldsymbol{F}}^{(l)} = \sigma\left(\left(\vec{\boldsymbol{F}}_{\mathcal{N}}^{(l-1)} \copyright \vec{\boldsymbol{F}}^{(l-1)}\right)\cdot \boldsymbol{W}^{(l)}\right) \quad (2)$$

where $\vec{\boldsymbol{F}}_{\mathcal{N}}^{(l)}$ is the feature representation of neighboring nodes. $\vec{\boldsymbol{F}}^{(l)}$ is the feature representation of all nodes. l means the lth aggregation layer (operation). The trainable model parameters w_r are assigned for $\forall r \in \mathcal{R}$ corresponding to different port types. © is the concatenation operation. $\vec{\boldsymbol{W}}^{(l)}$ are trainable model parameters and $\sigma(\cdot)$ is LeakyReLU function. Essentially, $\vec{\boldsymbol{W}}^{(l)}$ and $\sigma(\cdot)$ form a typical fully-connected (FC) layer to extract features from neighbor nodes and the node itself. For a node $v \in \mathcal{V}$, its degree is defined as $d(v) = \sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}} w_r h(v,e)_r$. For an edge $e \in \mathcal{E}$, its degree is defined as $\delta(e) = \sum_{v\in\mathcal{V}}\sum_{r\in\mathcal{R}} h(v,e)_r$. Further, $\mathbf{D}_e$ and $\mathbf{D}_v$ denote the diagonal matrices of the edge degrees and the node degrees, respectively. Stacking multiple aggregation layers can lead to numerical instabilities and increase the risk of exploding or vanishing gradients. To mitigate these issues, the inverses of $\mathbf{D}_e$ and $\mathbf{D}_v$ are employed to normalize the contributions of edges and nodes. As shown in Fig. 5, $\sum_{r\in\mathcal{R}} \vec{H}_r^{\top}$ is used to aggregate information from neighbor nodes to the hyperedge itself. $\sum_{r\in\mathcal{R}} w_r \vec{\boldsymbol{H}}_r$ is adopted to aggregate information from neighbor hyperedges to the node itself. Thus, through the node-hyperedge-node aggregation, we can efficiently extract the high-order correlation on the hypergraph. The aggregation, as shown in (1) and (2), is recursively and sequentially performed several times, then each node and its neighbor topology (subcircuit) is encoded as a feature vector.


aggregate from nodes to the hyperedge
aggregate from hyperedges to the node
Aggregation
Concat
Sum
MLP
C
Σ
Fig. 5. Our HGNNs.

HGNNs Model: We acquire the initial node feature vector as shown in Fig. 4 for each node and stack them to the feature matrix $\vec{\boldsymbol{F}}^{(0)}$, which serves as the input to the model. After processing through L aggregation layers, we obtain $\vec{\boldsymbol{F}}^{(L)}$. This matrix contains $|V|$ vectors, where $|V|$ is the node number and $\vec{\boldsymbol{f}}_i^{(L)}$ represents the embedding of the ith node at the Lth layer.

In subgraph recognition task, model performance is sensitive to network depth. With L aggregation layers, the model captures L-hop graph neighborhood structure around center nodes or anchors. This implies that larger graphs require more aggregation layers to ensure nodes can fully absorb comprehensive information from the entire graph. Considering the varying sizes of the graphs, we integrate skip connections between the first $L-1$ aggregation layers and the Lth aggregation layer to enhance the model scalability. Consequently, the final feature representation of the ith node, $\vec{\boldsymbol{f}}_i$, is a concatenation of the initial feature and all aggregation layers' outputs, that is $\vec{\boldsymbol{f}}_i = \copyright_{l=0}^{L}\vec{\boldsymbol{f}}_i^{(l)}$, as shown in Fig. 5. After obtaining the final feature representation for each node, we sum all the node embeddings within a graph, then pass the result through a fully connected layer to obtain the graph embedding, denoted as $\vec{z}$.

We find that the incorporation of skip connections enables the model to be extended to deeper layers while maintaining performance on small-radius graphs, thus effectively handling graphs of varying sizes and structures. Additionally, to prevent overfitting, we add a dropout layer with a rate of 0.2 after each aggregation layer, which proves beneficial for model performance.

Search Space Pruning: Up until this point, HGNNs have been employed to encode a subgraph (subcircuit) into a vector within the embedding space. Subsequently, a coarse subcircuit recognition technique is introduced within the embedding space, leveraging geometric relationships between the query circuit and all candidate subcircuits within the target circuit. As subgraph relationships naturally induce a partial ordering among subgraphs, it is beneficial to define an embedding space that preserves these subgraph relations. To accomplish this, we postulate that if a query graph is a subcircuit of the candidate graph, the embedding vector of the former should be positioned toward the lower left relative to the latter, as

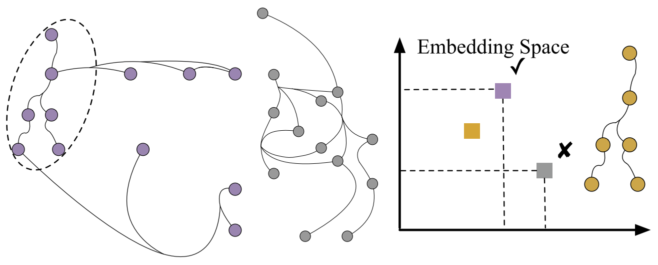


Fig. 6. Order embedding: golden is the query graph, purple and gray are the candidate graphs. The golden is a subgraph of the purple. Embedding vector associated with the former is positioned toward the lower left in relation to the purple.

depicted Fig. 6. The geometric relation can be represented as

$$\vec{z}_q \leq \vec{z}_c \text{ if and only if } \mathcal{G}_q \subseteq \mathcal{G}_c \tag{3}$$

where $\vec{z}_q$ and $\vec{z}_c$ denote that embedding feature vectors of the query circuit $\mathcal{G}_q$ and candidate circuit $\mathcal{G}_c$. $\vec{z}_q \leq \vec{z}_c$ mean each element in $\vec{z}_q$ is not greater than the counterpart in $\vec{z}_c$. $\mathcal{G}_q \subseteq \mathcal{G}_c$ means $\mathcal{G}_q$ is a subcircuit of $\mathcal{G}_c$.

To achieve geometric relation for subcircuit recognition, an order embedding-based loss function [21] is utilized as follows to train our HGNNs:

$$\mathcal{L}(\vec{z}_q, \vec{z}_c) = \sum_{(\vec{z}_q,\vec{z}_c)\in\mathcal{P}} E(\vec{z}_q, \vec{z}_c) + \sum_{(\vec{z}_q,\vec{z}_c)\in\mathcal{N}} \max\{0, \alpha - E(\vec{z}_q, \vec{z}_c)\} \tag{4}$$

where

$$E(\vec{z}_q, \vec{z}_c) = ||\max\{\vec{\mathbf{0}}, \vec{z}_q - \vec{z}_c\}||_2^2. \tag{5}$$

$\mathcal{P}$ denotes the set of positive samples in minibatch, where $\mathcal{G}_q$ is a subgraph of $\mathcal{G}_c$. While $\mathcal{N}$ denotes the set of negative samples, where $\mathcal{G}_q$ is not a subgraph of $\mathcal{G}_c$. α is a hyperparameter. By using order embeddings, we can ensure that the geometric relation within the embedding space can be used to perform subcircuit recognition, as shown in Fig. 6. According to (4), for positive samples, $E(\vec{z}_q, \vec{z}_c)$ is minimized when the embedding $\vec{z}_q$ is smaller than the embedding $\vec{z}_c$. For negative samples, $E(\vec{z}_q, \vec{z}_c)$ should be at least a parameter α to yield zero loss. When inference, $E(\vec{z}_q, \vec{z}_c)$ determines the subgraph relationship. If $E(\vec{z}_q, \vec{z}_c) \leq \alpha$, then $\mathcal{G}_q$ is considered a subgraph of $\mathcal{G}_c$; otherwise, it is not.

The model serves the purpose of pruning the candidate graph set, yet it does not attain 100% accuracy. In the event that the model wrongly identifies a negative sample as positive, such errors can be rectified during the subsequent RM procedure. However, if the model misclassifies a positive sample as negative, the exclusion of these samples from further matching processes leads to an irrecoverable loss of potentially correct matches. Consequently, the model's recall rate, representing its capability to accurately identify all positive samples, assumes paramount importance in preserving the integrity of the matching process. To enhance the recall rate, we adopt a relaxation strategy by allowing the recognition of a subgraph relationship when $E(\vec{z}_q, \vec{z}_c) \leq T$, where T is a new threshold set higher than α. While this adjustment may potentially reduce the precision rate, it guarantees that no valid matching graphs are overlooked, thereby ultimately improving the overall accuracy of the matching results.

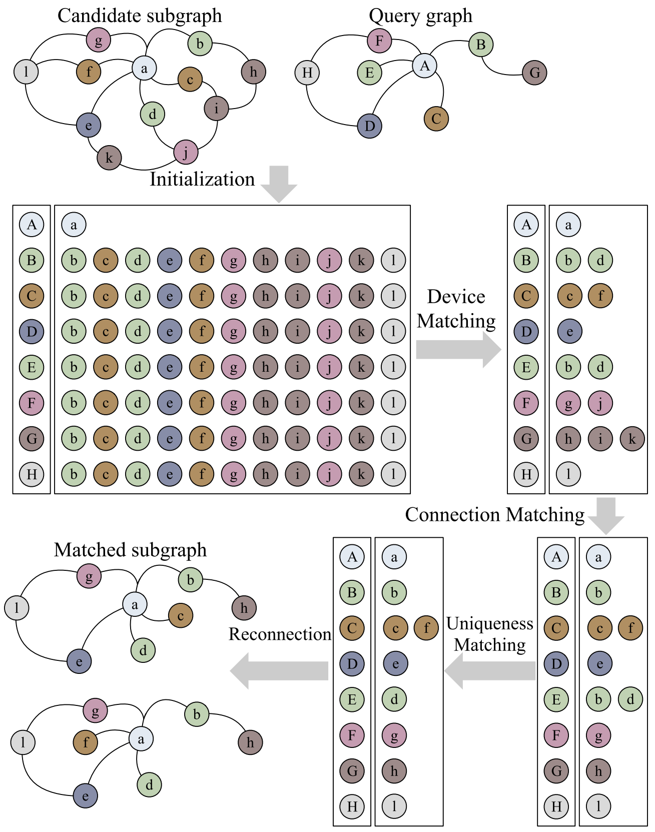


Fig. 7. RM for 2 overlapped subgraphs.

C. Radial Matching

In the second phase, we present a RM technique to facilitate exact matching between nets and devices in the query graph $\mathcal{G}_q$ and each candidate graph $\mathcal{G}_c$ obtained from the previous stage. This method is purposefully tailored to address the subgraph matching challenge in large-scale analog circuits, capitalizing on exploiting device attributes and connectivity relationships. Diverging from conventional DFS-based search approaches, our BFS-based RM methodology identifies all potentially matched subgraphs in a single traversal, eliminating the need for backtracking.

Simultaneously, our approach incorporates a progressive matching and error-checking mechanism, swiftly returning a False result if no subgraph is discovered. This strategy effectively strikes a balance between runtime and accuracy. Besides, our method can be expanded to fuzzy matching scenarios by accommodating the error tolerance.

For every candidate graph $\mathcal{G}_c$, the RM technique is employed to perform a step-by-step matching between each node in $\mathcal{G}_q$ and $\mathcal{G}_c$. The overall flow, as depicted in Fig. 7, encompasses three essential steps: 1) initialization; 2) matching; and 3) reconnection. Matching includes the following

Algorithm 1 RM

```
Input: The candidate graph $\mathcal{G}_c$, the query graph $\mathcal{G}_q$ and their initial matching table;
Output: The updated matching table.
1: For all rows in the matching table, perform device attribute matching (Algorithm 2);
2: For all rows in the matching table, perform connection relationship matching (Algorithm 3);
3: repeat
4:     For all rows in the matching table, perform uniqueness-based matching;
5:     For all updated rows, perform connection relationship matching;
6: until Matching table is not changed (Algorithm 4).
```

Algorithm 2 Device Attribute Matching

```
Input: The candidate graph $\mathcal{G}_c$, the query graph $\mathcal{G}_q$ and their initial matching table;
Output: The updated matching table.
1: for each row in the matching table do
2:     Check each candidate's device type and the number of connected nets and device neighbors.
3:     if a candidate's device type does not match the index node, or it has fewer nets or neighbors than the index node. then remove this candidate from the row.
4:     end if
5: end for
```

substeps: 1) device attribute matching; 2) connection relationship matching; and 3) uniqueness-based matching (and its loop framework).

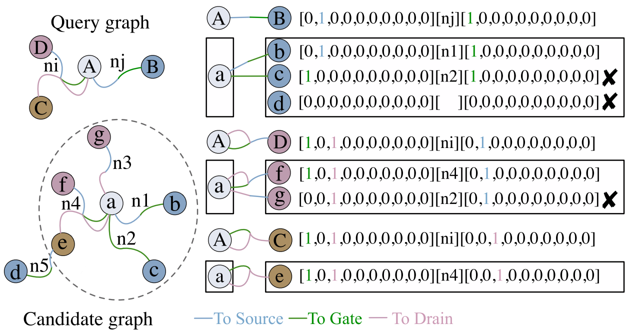
Initialization: In the initialization step, the matching table is set up with all nodes of $\mathcal{G}_q$ forming the row indices for the entire table, named index nodes, while all nodes from $\mathcal{G}_c$ are initially listed as the candidates in each row. The structure of the matching table is depicted in Fig. 7. The matching table indicates which nodes in $\mathcal{G}_q$ correspond to which nodes in $\mathcal{G}_c$, the former refers to the index nodes, while the latter refers to the candidates. We will progressively apply the structural and attribute information of the circuits from $\mathcal{G}_q$ and $\mathcal{G}_c$ to refine the candidates. Consequently, the number of candidates in the matching table will gradually decrease, eventually leaving only a small subset that fully matches the index nodes.

Following the definition in Section III-B, we align the center node of $\mathcal{G}_c$, as an anchor, with the center node u of the query graph $\mathcal{G}_q$, designating it as the candidate for u. Afterwards, we perform a three substep matching process outlined in Algorithm 1. To maintain efficiency, the algorithm halts immediately if any row in the matching table becomes empty, indicating that the index node in $\mathcal{G}_q$ cannot match any node in $\mathcal{G}_c$. Consequently, no matching subgraph is found, and the process moves to the next graph.

Device Attribute Matching: During the device attribute matching step, based on our extracted node features, we systematically traverse the matching table to establish connections between the device attributes of each index node and its corresponding candidates. As described in Algorithm 2, if a candidate exhibits disparities in device type or possesses fewer connected nets and nodes than its index node, it is promptly eliminated from consideration within the respective row. The check can be efficiently achieved by comparing the previously extracted feature vectors. The first update of the matching table in Fig. 7 illustrates this process, where the different colors of the nodes represent different types of devices.

Algorithm 3 Connection Relationship Matching

```
Input: The candidate graph $\mathcal{G}_c$, the query graph $\mathcal{G}_q$ and their matching table;
Output: The updated matching table.
1: for each layer from the center $u$ of $\mathcal{G}_q$ do to the edge
2:     for index nodes in the layer do
3:         Validate the connection relationships between the index node's row and its neighbors' row and update their candidates.
4:     end for
5: end for
```


Query graph
D
ni
A
nj
B
C
g
n3
f
n4
a
n1
b
n2
e
d
n5
c
Candidate graph
A
B
[0,1,0,0,0,0,0,0,0,0][nj][1,0,0,0,0,0,0,0,0,0]
a
b
[0,1,0,0,0,0,0,0,0,0][n1][1,0,0,0,0,0,0,0,0,0]
c
[1,0,0,0,0,0,0,0,0,0][n2][1,0,0,0,0,0,0,0,0,0] ✗
d
[0,0,0,0,0,0,0,0,0,0][ ][0,0,0,0,0,0,0,0,0,0] ✗
A
D
[1,0,1,0,0,0,0,0,0,0][ni][0,1,0,0,0,0,0,0,0,0]
a
f
[1,0,1,0,0,0,0,0,0,0][n4][0,1,0,0,0,0,0,0,0,0]
g
[0,0,1,0,0,0,0,0,0,0][n2][0,1,0,0,0,0,0,0,0,0] ✗
A
C
[1,0,1,0,0,0,0,0,0,0][ni][0,0,1,0,0,0,0,0,0,0]
a
e
[1,0,1,0,0,0,0,0,0,0][n4][0,0,1,0,0,0,0,0,0,0]
—To Source —To Gate — To Drain
Fig. 8. Connection relationship matching.

Connection Relationship Matching: During the connection relationship matching step, the connection relationships within $\mathcal{G}_q$ are meticulously identified. The connection relationship between any two nodes in a graph is defined as a list of triplets known as connection information. Each triplet in the list explicitly signifies a physical connection between one node and another via a net. The triplet takes the form of [connection of node1 to net][net][connection of net to node2]. Both the left and right items in the triplet are represented by 10-D vectors, which are the same as our node feature, as shown in Fig. 4, indicating the devices' ports connected to the net. The middle element in the triplet denotes the net name. Note that if the connection information between two nodes is empty, it indicates that they are not connected. Conversely, if the connection information between two nodes contains multiple triplets, it signifies that they are interconnected through multiple nets. This scenario is particularly common in multiport devices, notably transistors. In this format, all interconnections in the graph $\mathcal{G}_q$ are encoded. We then continue matching each node's candidates with all its neighboring nodes' candidates until all nodes in the corresponding matching table row have been processed, as described in Algorithm 3.

An exemplary case is depicted in Fig. 8, showcasing the connection relationship matching process for node A.

Algorithm 4 Uniqueness-Based Matching & Loop Framework

```
Input: The candidate graph $\mathcal{G}_c$, the query graph $\mathcal{G}_q$ and their matching table;
Output: The updated matching table.
1: repeat
2:     for each row in the matching table do
3:         According to number of candidate(s), establish single candidate set.
4:     end for
5:     for each row with multiple candidates do
6:         Eliminate candidates already existing within the single candidate set. Update the rows accordingly.
7:     end for
8:     For all updated rows, perform connection relationship matching;
9: until Matching table is not changed.
```

Specifically, the gate and drain of MOS A are connected to the source of MOS D via the net ni. Hence, the connection information between A and D can be represented as [1, 0, 1, 0, 0, 0, 0, 0, 0, 0] [ni] [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]. Subsequently, we encode all interconnections within $\mathcal{G}_q$ and validate the connection information of rows pertaining to A's neighboring nodes B, C, and D in a pairwise manner. Taking node A and B as an example, only candidate b successfully passes the validation process, while candidates c and d fail and are consequently eliminated. Following the completion of connection matching between A and B, C, D, the connection relationship matching for node A concludes. In a similar fashion, we traverse all the nodes that require processing, systematically updating the rows of the matching table correspondingly.

Uniqueness-Based Matching: During the uniqueness-based matching step and its loop framework described in Algorithm 4, we systematically traverse the matching table to identify all rows exclusively containing a single candidate, thereby establishing a single candidate set.

For each row containing multiple candidates, we eliminate those already existing within the single candidate set. This process involves updating the rows accordingly, as depicted in Fig. 7. For instance, in the given example, node b is removed from the candidate matching list of node E since it has already been matched with node B. Subsequently, we locally reapply Algorithm 3 to the rows where the candidates have changed, attempting to further narrow down the candidate lists. If the candidates are reduced, we use this updated matching table as input and reinvoke the uniqueness-based matching. This recursive structure will continue to execute until the matching table no longer changes. At this point, our RM step concludes, and the final reconnection step is about to begin.

Reconnection: Upon the completion of iterations in Algorithm 1, if the matching table remains unaltered, it indicates that a finalized matching table has been achieved. At this stage, we proceed to extract each candidate node and use the original topological information from $\mathcal{G}_q$ alongside the stored connection data among candidates to reconstruct one or several subgraphs of $\mathcal{G}_c$. In the event that each row exclusively contains a single candidate, it signifies the presence of a unique subgraph within $\mathcal{G}_c$ that exactly matches $\mathcal{G}_q$. Utilizing the information from $\mathcal{G}_c$ and $\mathcal{G}_q$, we reconstruct the candidate nodes and their corresponding connection into a graph structure, which is subsequently appended to the list of matched subcircuits. Conversely, if a row with multiple candidates exists within each row, it implies the existence of multiple subgraphs within $\mathcal{G}_c$ that match $\mathcal{G}_q$ and potentially overlap with one another. In this case, we proceed to reconstruct all possible graphs and add them to the list of matched subcircuits.

TABLE II
TYPICAL VALUES TABLE

Task Attributes	
Target Size (# of devices)	50k-1000k
Query Size (# of devices)	5-30
# of Isomorphic	0-10k
Preparation Time	
Network Training Time	5-8h
Netlist Import & Parse Time	1-10s

D. Framework Extension Discussion

While our SMART framework primarily targets flattened post-layout netlists, our proposed approach demonstrates excellent scalability and can be readily extended to handle hierarchical circuit subgraph identification through systematic modifications. During the transformation of hierarchical circuit netlists into hypergraphs, we recursively parse and record each device's hierarchical level information, thereby enabling the augmentation of device node properties with corresponding hierarchical labels during hypergraph construction.

Following subgraph searching, to analyze the correlation between recognized subgraphs and their corresponding locations in the originally structured netlist, we can leverage the embedded hierarchical information within the labels to efficiently pinpoint their precise positions in the source netlist hierarchy. This hierarchical mapping capability enables rapid and accurate traceability between the identified subgraphs and their original circuit contexts.

IV. EXPERIMENTAL RESULTS

A. Experimental Setting

The analog circuit topology synthesis framework [32] is used to generate large-scale analog circuit netlists (SPICE format) as our benchmarks. The SPICE netlists are parsed by Ngspice [33], an open-source SPICE simulator. We develop our HGNNs model with PyG [34], which is a library built based on PyTorch [35]. Our RM is implemented in Python with Networkx library [36]. Our experiments are conducted on a Linux machine with 80 cores (2.70 GHz) and 512-G RAM and NVIDIA Tesla A100 GPU with 80-GB memory.

Some typical values from this experiment are shown in Table II. The task attributes represent the typical range for the

TABLE III
Runtime and Accuracy Comparison of Subcircuit Matching Methods

# of Device		# of Isomorphic	VF3		Partition+VF3		HGNNs+VF3		RM		SMART (HGNNs+RM)	
Target	Query		RT (s)	Acc.	RT (s)	Acc.	RT (s)	Acc.	RT (s)	Acc.	RT (s)	Acc.
Synthetic Circuits												
58k	10	6	19.57	100%	9.71	100%	2.85	100%	2.82	100%	**0.27**	**100%**
200k	10	742	35.78	100%	26.66	100%	6.31	100%	15.71	100%	**1.09**	**100%**
400k	10	1707	139.20	100%	93.73	100%	18.71	99.7%	62.61	**100%**	**5.18**	99.7%
600k	10	3387	594.47	100%	253.09	100%	50.80	99.4%	179.73	**100%**	**11.94**	99.4%
1000k	10	4389	2351.66	100%	973.47	100%	88.75	99.2%	375.52	**100%**	**17.42**	99.2%
Real-world Circuits												
288	10	1	**0.011**	**100%**	0.024	100%	0.062	100%	0.034	100%	0.070	**100%**
10.7k	10	2	5.24	100%	4.27	100%	0.42	100%	2.82	100%	**0.24**	**100%**
153.7k	10	307	35.04	100%	30.57	100%	6.45	99.7%	5.72	**100%**	**1.09**	99.7%
515.5k	10	446	158.87	100%	85.74	100%	13.21	99.8%	20.48	**100%**	**2.64**	99.8%

subcircuit matching problem in large-scale analog circuits. The netlists we use for the experiment are post-layout verification netlists that have been flattened and contain no hierarchical information. This lack of hierarchy is precisely what makes this problem challenging. # of Isomorphic means in the target circuit, the golden number of subcircuits isomorphic to the query circuit. The preparation time indicates the time consumption not included in the matching task.

B. Dataset

Data Augmentation: Limited training datasets often lead to poor generalization in analog circuit recognition. Data augmentation has proven to be an effective solution to this challenge [16]. We expand the analog circuit topology synthesis framework [32] to generate synthetic circuits as the dataset. The original framework defines a library of fundamental building blocks (e.g., current mirrors, cascade stages, differential pairs) in analog circuits. Reinforcement learning is employed to guide their connection into legal analog circuits. However, the original framework has the issue of a lack of diversity, only containing limited analog components. We first expand the original building block library by incorporating passive components, more analog circuit variants, and common digital gates to better support analog/mixed-signal (AMS) circuits. Additionally, we generate circuits sequentially and modify the reward function considering similarity between the current circuit and existing ones to further increase diversity.

Dataset Partition: Ultimately, we generate 2000 unique large-scale analog circuits containing 500–800 devices each. We partition the dataset into training and test sets with an 8:2 ratio for model training and validation. Additionally, we verify the efficiency of our proposed method using some large circuits (containing over 50 k devices) generated by the framework and evaluate the generalization capability using some real-world circuits, including ITC'17 benchmark circuits [37] and industrial designs.

Batch Construction: To train the model, we adopt a mini-batch approach, where each batch comprises both positive and negative samples. For the positive sample, we randomly select a base circuit from the training set and designate a center device within it. Subsequently, we define the target circuit as the k-hop circuit surrounding this center device. To construct the query circuit, we initiate a random walk starting from the center device within the target circuit, ensuring that the query circuit also possesses a radius of k. This systematic procedure guarantees that the query circuit is isomorphic to subcircuits of the target circuit.

Regarding the negative sample, we follow a similar method to randomly select a circuit from the training set and extract a target circuit. Subsequently, we choose a distinct circuit and center device to generate the query circuit. Crucially, we verify that the query circuit is not isomorphic to any subcircuit of the target circuit by VF3 [20], [38], ensuring that it represents a genuine negative sample. VF3 is a state-of-the-art general subgraph isomorphism framework. Thus, we use it to obtain the golden result about subcircuits isomorphic to the query circuit and obtain matches between interconnections and devices in the query circuit and target circuits. Maintaining an appropriate balance between positive and negative samples, each minibatch is configured to have a ratio of 1:3 for positive to negative samples, facilitating robust training of the model.

C. Model Training

A progressive training strategy is employed during the training stage to enhance the model's versatility in effectively processing graphs of varying sizes and structures. Initially, the model is trained using 1-hop graphs sampled from the training set, utilizing the Adam optimizer with an initial learning rate of 10^{-4}. Once the model's performance stabilizes, the graph radius progressively increases to 6. The batch size is set to 128. To improve training efficiency, we maintain a record of sampled graph pairs to avoid duplicates and perform model training and graph sampling in parallel for each batch. The learning rate is dynamically adjusted using a cosine annealing schedule, with a reset after each radius increment.

D. Overall Comparison in Subgraph Matching

VF3 [20], [38] is used as a baseline to compare with our RM and SMART (HGNNs+RM). The comparison results are shown in Table III, where Partition+VF3 refers to partitioning the target circuit to be all potential subcircuits before using

VF3, including both the time for partitioning and subsequent VF3 algorithm execution time. HGNNs+VF3/RM is a two-stage method, where our HGNNs prune the research space, and then VF3/RM is used to perform matching, including both the HGNNs runtime and the VF3/RM algorithm execution time.

To thoroughly evaluate SMART's performance characteristics across tasks of varying scales, we leveraged the analog circuit topology synthesis framework mentioned in Section IV-B to generate five new synthetic circuits of different scales to form the first group of target graphs. Also, to evaluate the SMART's practical applicability and generalization ability, we selected four post-layout netlists from real-world circuits of diverse types and scales to form the second group of target graphs, including VCO (288 devices), SerDes (10.7k devices), SoC1 (153.7k devices) and SoC2 (515.5k devices).

We chose query graphs of size 10 from each group. Based on our experience, when the query size is less than 7, the complexity of each subtask is insufficient to evaluate performance differences and accuracy effectively. Conversely, when the query size exceeds 15, the number of isomorphic subcircuits decreases sharply, often tending toward one. Therefore, selecting a query size of ten ensures that the subtasks are sufficiently complex and that a significant number of isomorphic instances are present, making this the most complex and suitable scenario for performance evaluation.

Synthetic Circuits: When dealing with topology-diverse synthetic circuits, our HGNNs can efficiently prune research space, significantly reducing runtime. Specifically, when the target circuit has fewer than 200k devices, our SMART can achieve 100% matching accuracy. As the circuit size expanded to 400k devices and the subgraph number further increased to 1700+, the execution time of VF3 and Partition+VF3 becomes huge. Yet, our RM method still maintains competitive speed. Unlike VF3's general-purpose graph matching approach, the superior runtime of RM stems from its circuit-specific pruning strategies that effectively reduce the search space while maintaining accuracy through complete exploration; at this point, the network's accuracy begins to slightly decline, which remains within an acceptable range. It can be observed that both VF3 and RM experience significant speed improvements after utilizing the network. Because of HGNNs, our SMART achieves nearly 100% matching accuracy when handling large-scale target circuits with complicated topology structures.

Real-World Circuits: When handling different real-world circuits, our SMART exhibits performance characteristics consistent with the synthetic circuit experiments. For small-scale conventional circuits, such as VCO, where the task complexity is limited, the overhead from circuit partitioning, matching table updating and HGNNs model deployment results in no significant runtime advantage for SMART and RM, an expected outcome for such tasks. When processing more sophisticated circuits like SerDes, our SMART achieves optimal runtime efficiency while maintaining perfect matching accuracy. In the case of SoC1 and SoC2, HGNNs demonstrate remarkable speed improvements with only one subcircuit miss, representing a minimal accuracy tradeoff.

Summary: Our SMART (HGNNs+RM) surpasses all other methods in runtime with negligible accuracy loss. In all cases where the accuracy is 100%, our RM method achieves the best speed, further demonstrating the superiority of our approach.

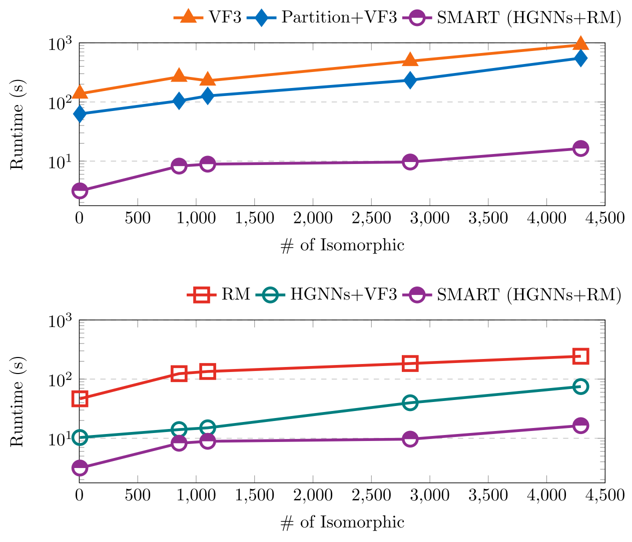


Fig. 9. # of subcircuit versus runtime.

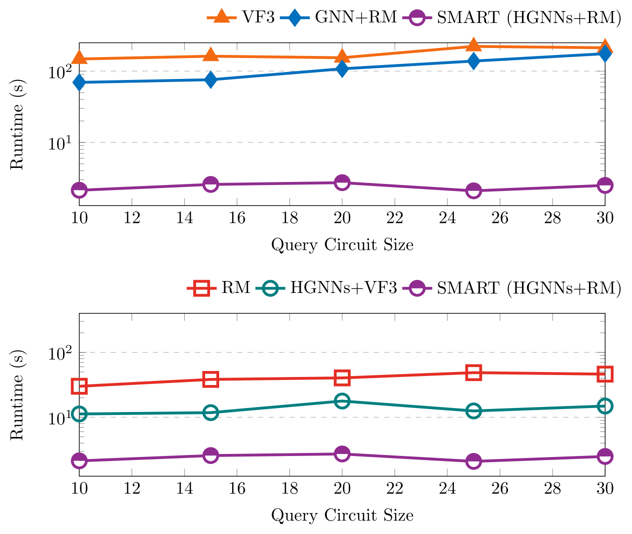


Fig. 10. Query circuit size versus runtime.

E. Runtime Comparison in Subgraph Matching

Figs. 9 and 10 present the detailed results of controlled variable testing, specifically focusing on the runtime for subgraph matching in typical large-scale circuit scenarios (circuit size: 600k devices). In Fig. 9, the variable considered is the number of isomorphic subcircuits (# of Isomorphic), while the size of the query circuit remains 10. In Fig. 10, the variable considered is the size of the query circuit (# of Query Device), while the number of isomorphic subcircuits remains 1. The results show our SMART can achieve excellent scalability.

F. Runtime Comparison for Small Patterns

To investigate the algorithm's robustness and performance boundaries, we conducted a series of experiments focusing on

TABLE IV
RUNTIME AND ACCURACY COMPARISON FOR SMALL PATTERNS

# of Device		# of Isomorphic	VF3		Partition+VF3		HGNNs+VF3		RM		SMART (HGNNs+RM)	
Target	Query		RT (s)	Acc.	RT (s)	Acc.	RT (s)	Acc.	RT (s)	Acc.	RT (s)	Acc.
600k	10	114	46.32	100%	6.45	100%	3.32	100%	5.24	100%	**0.40**	**100%**
600k	8	234	60.26	100%	9.01	100%	4.30	100%	7.28	100%	**0.88**	**100%**
600k	6	2941	115.74	100%	54.93	100%	24.23	99.7%	55.58	100%	**9.49**	**100%**
600k	4	10698	241.18	100%	108.55	100%	34.99	99.4%	98.97	100%	**21.66**	**100%**
600k	2	29876	400.54	100%	120.75	100%	30.76	99.2%	65.10	100%	**23.10**	**100%**

TABLE V
PRECISION (%) AND INFERENCE TIME (MS) OF GRAPH LEARNING MODELS IN OUR GRAPH EMBEDDING STAGE AT DIFFERENT RADII (RECALL=100%)

Radius	NeuroMatch [21]		R-SAGE [17]		HyperGNN [39]		HGAT [40]		Ours	
	Precision	Inference	Precision	Inference	Precision	Inference	Precision	Inference	Precision	Inference
1	89.7	**35.2**	94.1	38.5	92.6	40.2	93.8	58.6	**94.6**	48.7
2	84.5	**38.6**	90.7	42.3	89.3	47.5	91.4	73.4	**91.9**	56.1
3	79.5	**43.5**	83.8	48.2	82.4	55.6	83.7	95.8	**85.1**	58.4
4	77.3	**58.9**	80.2	65.4	80.7	66.3	81.2	122.3	**83.9**	70.9
5	73.5	**65.3**	77.1	72.8	78.6	79.8	80.3	149.6	**84.0**	85.7
6	68.1	**74.8**	73.5	83.5	74.9	96.7	76.2	183.5	**80.8**	103.0
Ave.	78.8	**52.7**	83.2	58.5	83.1	64.4	84.4	113.9	**86.7**	70.5

scenarios where query size contains fewer than ten devices, as shown in Table IV. The experimental setup maintained consistent variables by utilizing a fixed target circuit of 600k devices and the same query circuit, while systematically reducing its # of Device to a minimum of two devices.

As query size decreased, # of Isomorphic gradually increased, showing explosive growth when reduced to four devices, leading to increased total runtime. Given this characteristic, we selected a representative case set to investigate how the explosive growth in subgraph count affects runtime: initially having only 100 subgraphs, then increasing a hundredfold as the query circuit decreased. However, since each subcircuit was smaller, individual task complexity also decreased, keeping total runtime within manageable limits. We observed that our neural network performed well with small circuits, achieving 100% accuracy while maintaining the fastest speed.

Notably, for our research domain, the typical query size of interest is 5–30 (as shown in Table II). When the size drops to 5, the practical value becomes limited. The cases with sizes 4–2 were included solely to examine the algorithm's performance and robustness under various conditions.

G. Ablation Study

We compare several sets of data to further demonstrate the rationality and effectiveness of our framework, as shown in Table III, Figs. 9 and 10.

When comparing Partition+VF3 with VF3, the speed increases by twofold or more, indicating that our Partition strategy is effective in reducing the task size. The performance of our RM, which also applies the Partition strategy, significantly surpasses VF3 in speed without any loss in accuracy. This superior performance can be attributed to two key factors: On the one hand, both RM and VF3 are combinatorial optimization algorithms, ensuring 100% accuracy in their results. On the other hand, RM more effectively utilizes the inherent topological information in circuits. By leveraging this circuit-specific knowledge, RM performs more efficient searches than the general-purpose VF3. This is extremely important as it not only ensures the excellent performance of SMART but also provides the best current option for scenarios requiring 100% accuracy. Comparing RM with HGNNs+RM, it can be seen that HGNNs achieve significant speedup while maintaining accuracy within an acceptable range, demonstrating the effectiveness of our framework in subgraph matching applications.

H. Graph Learning Model Evaluation in Embedding Stage

We select the following learning methods as our graph learning baseline models: NeuroMatch [21], R-SAGE [17], HyperGNN [39], and HGAT [40]. All models are configured with 8 aggregation layers and share the same order embedding loss function, with α set to 0.2. NeuroMatch and R-SAGE use directed multigraph representation shown in Fig. 2(b), while the remaining methods use hypergraph representation shown in Fig. 2(d). They are used in our proposed graph embedding stage as shown in Fig. 3.

In order to evaluate their accuracy, we use Recall and Precision, which are defined as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6}$$

where TP represents true positives and FN represents false negatives. FP denotes false positives. Recall measures the model's ability to identify all positive instances. A 100% recall indicates that the model can identify all positive instances. Precision measures the model's accuracy in making positive predictions. It reflects the model's capability to exclude negative samples while maintaining a high recall rate accurately.

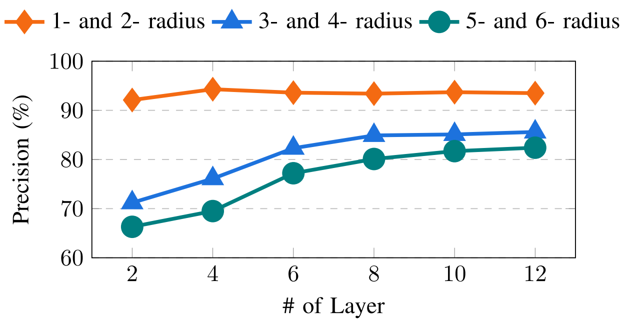


Fig. 11. Precision (%) for different layers.

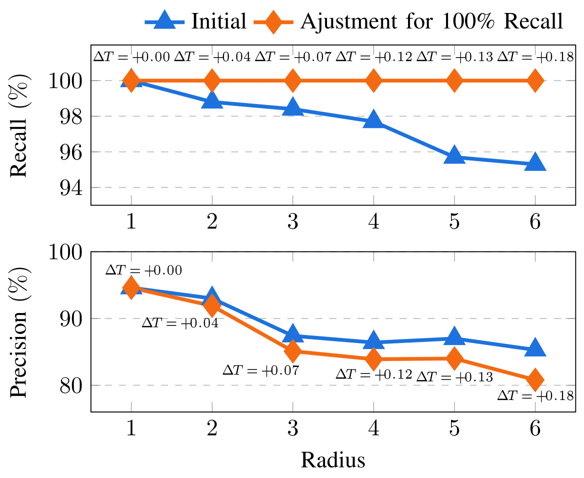


Fig. 12. Recall/Precision (%) versus threshold.

After the model is well trained, we adopt the relaxation strategy to adjust the threshold T so that all recalls reach 100% and use the precision rate to evaluate model performance. Table V shows the performance and inference time (batch size = 512) of graph learning models at different radii. Our HGNNs can outperform all baseline models with the highest precision among different circuit sizes. NeuroMatch [21] employs GraphSAGE[41] for information aggregation and achieves fast inference speed. However, it exhibits a significant performance drop as the radius increases, suggesting traditional GNNs may not capture complex higher-order information effectively in large-scale analog circuits. In the directed multigraph representation shown in Fig. 2(b), different edge colors represent port types (such as drain and source). R-SAGE [17] extends GraphSAGE by introducing port-specific edge weights in the message aggregation process. Similar to our approach, R-SAGE incorporates port types during message passing and demonstrates improved performance compared to NeuroMatch, indicating that explicit consideration of port types is beneficial for subgraph matching tasks. Hypergraph-based methods like HyperGNN [39] and HGAT [40] better preserve circuit structural information through hypergraph representation, showing less decline at large-scale circuits but require longer inference time. Though our method does not demonstrate the advantage in terms of inference runtime, its superior accuracy justifies its effectiveness, particularly considering that model inference time accounts for a relatively small portion of the entire workflow.

We test the effect of the aggregation layer number L and change it from 2 to 12. We categorize the graphs into three classes based on their radii to investigate the impact of the aggregation layer number with different circuit sizes. Fig. 11 shows that for circuits with a given radius, when the aggregation layer number is not enough to cover the circuit radius, the performance improves quickly as the number of aggregation layers increases. Once the number of layers reaches the circuit radius, the performance continues to improve slowly with increasing aggregation layer number and eventually approaches convergence. However, we observe that despite the inclusion of skip connections in the network, the performance of 1- and 2-radius graphs still exhibits a slight decline when the number of layers increases. As the size of the query circuit is relatively small in the practical application of this work (radius $\leq$ 4), we set the number of layers in our model to 8 to achieve good performance and alleviate the potential over-smoothing issue.

We evaluated the threshold T impact. The initial threshold is $T = \alpha = 0.2$. Fig. 12 shows how the model precision and recall change at different radii as the threshold is adjusted. As the radius increases, a larger threshold adjustment is needed to maintain 100% recall. The space between the blue and orange lines demonstrates the effect of threshold adjustments on recall and precision. It shows that we can achieve an increase in recall with a tolerable decrease in precision.

V. Conclusion

In this article, we propose SMART, a graph learning-boosted subcircuit matching framework for large-scale analog circuits. We employ customized HGNNs to transform the circuit topology into an embedding space. Subsequently, coarse subcircuit recognition is executed directly. Then, a RM method is proposed to perform fine recognition and matching. The experimental results show our SMART surpasses the performance of state-of-the-art search-based method, VF3, and the learning-based approach, NeuroMatch.

References

[1] M. Ohlrich, C. Ebeling, E. Ginting, and L. Sather, “Subgemini: Identifying subcircuits using a fast subgraph isomorphism algorithm,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 1993, pp. 31–37.
[2] N. Rubanov, “SubIslands: The probabilistic match assignment algorithm for subcircuit recognition,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 1, pp. 26–38, Jan. 2003.
[3] H. Graeb and M. Leibl, “Learning from the implicit functional hierarchy in an analog netlist,” in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, 2023, pp. 93–100.
[4] Z. Wu, I. Song, and I. Savidis, “Hybrid utilization of subgraph isomorphism and relational graph convolutional networks for analog functional grouping annotation,” in *Proc. ACM/IEEE Workshop Mach. Learn. CAD (MLCAD)*, 2023, pp. 1–6.
[5] K. Kunal, J. Poojary, T. Dhar, M. Madhusudan, R. Harjani, and S. S. Sapatnekar, “A general approach for identifying hierarchical symmetry constraints for analog circuit layout,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–8.
[6] K. Kunal et al., “GNN-based hierarchical annotation for analog circuits,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 9, pp. 2801–2814, Sep. 2023.

[7] H. Chen, K. Zhu, M. Liu, X. Tang, N. Sun, and D. Z. Pan, “Universal symmetry constraint extraction for analog and mixed-signal circuits with graph neural networks,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 1243–1248.
[8] M. Su et al., “Subgraph matching based reference placement for PCB designs: Late breaking results,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2022, pp. 1400–1401.
[9] K. Kunal et al., “GANA: Graph convolutional network based automated netlist annotation for analog circuits,” in *Proc. Design, Autom. Test Europe (DATE)*, 2020, pp. 55–60.
[10] M. Neuner, I. Abel, and H. Graeb, “Library-free structure recognition for analog circuits,” in *Proc. IEEE/ACM Design, Autom. Test Europe (DATE)*, 2021, pp. 1366–1371.
[11] J. R. Ullmann, “An algorithm for subgraph isomorphism,” *J. ACM*, vol. 23, no. 1, pp. 31–42, 1976.
[12] J. L. Gross, J. Yellen, and M. Anderson, *Graph Theory and its Applications*. Boca Raton, FL, USA: Chapman Hall/CRC, 2018.
[13] T. Bücher, L. Alrahis, G. Paim, S. Bampi, O. Sinanoglu, and H. Amrouch, “AppGNN: Approximation-aware functional reverse engineering using graph neural networks,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2022, pp. 1–9.
[14] L. Alrahis et al., “GNN-RE: Graph neural networks for reverse engineering of gate-level netlists,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 8, pp. 2435–2448, Aug. 2022.
[15] Z. He, Z. Wang, C. Bai, H. Yang, and B. Yu, “Graph learning-based arithmetic block identification,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2021, pp. 1–8.
[16] A. Deeb et al., “A graph attention network based system for robust analog circuits’ structure recognition involving a novel data augmentation technique,” *IEEE Access*, vol. 12, pp. 29308–29344, 2024.
[17] Z. Wu and I. Savidis, “Comparative analysis of graph isomorphism and graph neural networks for analog hierarchy labeling,” in *Proc. IEEE Int. Symp. Qual. Electron. Design (ISQED)*, 2024, pp. 1–7.
[18] N. Rubanov, “A high-performance subcircuit recognition method based on the nonlinear graph optimization,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 11, pp. 2353–2363, Nov. 2006.
[19] H.-Y. Su, C.-H. Hsu, and Y.-L. Li, “SubHunter: A high-performance and scalable sub-circuit recognition method with prüfer-encoding,” in *Proc. IEEE/ACM Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2015, pp. 1583–1586.
[20] V. Carletti, P. Foggia, A. Saggese, and M. Vento, “Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 804–818, Apr. 2018.
[21] Z. Lou, J. You, C. Wen, A. Canedo, and J. Leskovec, “Neural subgraph matching,” 2020, *arXiv:2007.03092*.
[22] K. Chen and G. G. E. Gielen, “Self-learning and transfer across topologies of constraints for analog/mixed-signal circuit layout synthesis,” in *Proc. IEEE/ACM Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2024, pp. 1–6.
[23] H. Ren, S. Nath, Y. Zhang, H. Chen, and M. Liu, “Why are graph neural networks effective for EDA problems?” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2022, pp. 1–8.
[24] T. Chen, G. L. Zhang, B. Yu, B. Li, and U. Schlichtmann, “Machine learning in advanced IC design: A methodological survey,” *IEEE Design Test*, vol. 40, no. 1, pp. 17–33, Feb. 2023.
[25] T. Chen, Q. Sun, and B. Yu, “Machine learning in nanometer AMS design-for-reliability,” in *Proc. IEEE Int. Conf. ASIC (ASICON)*, 2021, pp. 1–4.
[26] Q. Ma, L. Xiao, Y.-C. Tam, and E. F. Y. Young, “Simultaneous handling of symmetry, common centroid, and general placement constraints,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 1, pp. 85–95, Jan. 2011.
[27] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu, “Analog IC aging-induced degradation estimation via heterogeneous graph convolutional networks,” in *Proc. IEEE/ACM Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2021, pp. 898–903.
[28] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu, “Deep H-GCN: Fast analog IC aging-induced degradation estimation,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 7, pp. 1990–2003, Jul. 2022.
[29] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, “Multilevel hypergraph partitioning: Application in VLSI domain,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 1997, pp. 526–529.
[30] T. Chen et al., “Wages: The worst transistor aging analysis for large-scale analog integrated circuits via domain generalization,” *ACM Trans. Design Autom. Electron. Syst.*, vol. 29, no. 5, pp. 1–23, 2024.
[31] B. Razavi, *Design of Analog CMOS Integrated Circuits*. New York, NY, USA: McGraw-Hill Educ., 2002.
[32] Z. Zhao and L. Zhang, “Analog integrated circuit topology synthesis with deep reinforcement learning,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5138–5151, Dec. 2022.
[33] H. Vogt, G. Atkinson, P. Nenzi, and D. Warning. “Ngspice user’s manual version 40 plus (ngspice release version).” 2023. [Online]. Available: https://ngspice.sourceforge.io/docs/ngspice-40-manual.pdf
[34] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch geometric,” 2019, *arXiv:1903.02428*.
[35] “Pytorch.” Accessed: Jan. 2024. [Online]. Available: https://pytorch.org/
[36] A. Hagberg, P. Swart, and D. S. Chult, “Exploring network structure, dynamics, and function using networkX,” Los Alamos Nat. Lab., Los Alamos, NM, USA, Rep. LA-UR-08-05495, 2008.
[37] S. Sunter and P. Sarson, “A/MS benchmark circuits for comparing fault simulation, DFT, and test generation methods,” in *Proc. IEEE Int. Test Conf. (ITC)*, 2017, pp. 1–7.
[38] “VF3.” 2024. [Online]. Available: https://pypi.org/project/vf3py/
[39] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, “Hypergraph neural networks,” in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2019, pp. 3558–3565.
[40] S. Bai, F. Zhang, and P. H. Torr, “Hypergraph convolution and hypergraph attention,” *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107637.
[41] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 30, 2017, pp. 1025–1035.

Jindong Tu received the B.S. degree from Shanghai University, Shanghai, China, in 2023. He is currently pursuing the Ph.D. degree with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhan), Shenzhen, China.

His research interests include analog circuit design automation and mixed-signal IC design.

Yapeng Li received the B.Eng. degree in microelectronics science and engineering from the South China University of Technology, Guangzhou, China, in 2023. He is currently pursuing the Ph.D. degree with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhan), Shenzhen, China.

His research interests include machine learning and algorithm optimization for analog EDA.

Pengjia Li received the B.Eng. degree in electronic science and technology from Southeast University, Nanjing, China. He is currently pursuing the Ph.D. degree with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhan), Shenzhen, China.

His research interests include machine learning for digital circuit design and deep learning accelerators.

Peng Xu received the B.S. degree from Central South University, Changsha, China, and the M.S. degree from the Harbin Institute of Technology, Harbin, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. B. Yu.

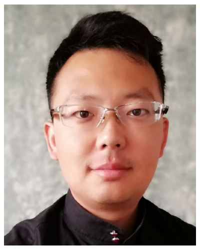
His research interests include machine learning for analog physical design and optimization in EDA problems.

Qianru Zhang received the M.Sc. degree in electrical and computer from the University of California at Irvine, Irvine, CA, USA, and the Ph.D. degree in microelectronics from Southeast University, Nanjing, China.

He is currently a Design-For-Reliability And Testability (DFX) Researcher with HiSilicon Technologies Company, Shenzhen, China. Her research interests include machine learning in VLSI DFX, failure mechanisms, and anomaly detection algorithms in industrial automation.

Sanping Wan received the M.Eng. degree in microelectronics and solid-state electronics from the Huazhong University of Science and Technology, Wuhan, China.

Since 2011, he has been engaged in VLSI design-for-reliability (DFR), design-for-testability (DFT), failure mechanism Research and DFR EDA tool and flow development with HiSilicon Technologies Company, Shenzhen, China.



Yongsheng Sun received the B.S. and M.S. degrees in microelectronics from the University of Electronic Science and Technology of China, Chengdu, China, in 2003 and 2006, respectively.

He is a Team Leader on Design For Reliability with Hisilicon, Shenzhen, China. In 2006, he joined Hisilicon Technologies Company Ltd. as a Quality and Reliability Engineer, responsible for supplier's quality and reliability management. Since 2011, his work is focused on design for reliability. The related areas are reliability mechanism study and modeling in advanced CMOS process technology, reliability simulation in analog circuit, and SoC level aging timing solution.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received 11 Best Paper Awards from ICCAD 2024, 2021, and 2013, IEEE TSM 2022, DATE 2022, ASPDAC 2021 and 2012, ICTAI 2019, Integration, the the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, six ICCAD/ISPD contest awards, IEEE CEDA Ernest S. Kuh Early Career Award in 2021, DAC Under-40 Innovator Award in 2024, and Hong Kong RGC Research Fellowship Scheme Award in 2024. He has served as the TPC Chair for ACM/IEEE Workshop on Machine Learning for CAD in 2019, International Symposium of EDA in 2025, and in many journal editorial boards and conference committees.

Tinghuan Chen (Member, IEEE) received the B.Eng. and M.Eng. degrees in electronics engineering from Southeast University, Nanjing, China, in 2014 and 2017, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong (Shenzhan), Shenzhen, China, in 2021.

He is currently an Assistant Professor in the School of Science and Engineering, The Chinese University of Hong Kong. His research interests include machine learning for EDA and deep learning accelerators.