Functional Matching of Logic Subgraphs: Beyond Structural Isomorphism

Ziyang Zheng Kezhi Li Zhengyuan Shi Qiang Xu
The Chinese University of Hong Kong
{zyzheng23,kzli24,zyzshi21,qxu}@cse.cuhk.edu.hk

Abstract

Subgraph matching in logic circuits is foundational for numerous Electronic Design Automation (EDA) applications, including datapath optimization, arithmetic verification, and hardware trojan detection. However, existing techniques rely primarily on structural graph isomorphism and thus fail to identify function-related subgraphs when synthesis transformations substantially alter circuit topology. To overcome this critical limitation, we introduce the concept of *functional subgraph matching*, a novel approach that identifies whether a given logic function is implicitly present within a larger circuit, irrespective of structural variations induced by synthesis or technology mapping. Specifically, we propose a two-stage multi-modal framework: (1) learning robust functional embeddings across AIG and post-mapping netlists for functional subgraph detection, and (2) identifying fuzzy boundaries using a graph segmentation approach. Evaluations on standard benchmarks (ITC99, OpenABCD, ForgeEDA) demonstrate significant performance improvements over existing structural methods, with average 93.8% accuracy in functional subgraph detection and a dice score of 91.3% in fuzzy boundary identification. The source code and implementation details can be found at *our repository*.

1 Introduction

Subgraph matching—identifying smaller graphs within larger ones—is a fundamental task in graph analysis, with pivotal applications spanning social network mining, bioinformatics, and Electronic Design Automation (EDA).

In the context of EDA, subgraph matching involves searching for specific circuit patterns embedded within larger circuits. This capability directly supports critical tasks such as circuit optimization, verification, and security analyses. For example, verifying complex arithmetic circuits like multipliers typically requires recognizing embedded small functional units (e.g., half-adders) within larger netlists, enabling algebraic simplifications and correctness proofs [1, 2]. Similarly, during template-based synthesis, accurately locating predefined subgraphs allows their replacement with highly optimized standard cells, thereby significantly improving power, performance, and area (PPA) metrics [3]. Moreover, subgraph matching also plays an essential role in hardware security by enabling the identification of potentially malicious substructures or "hardware trojans"—anomalous subcircuits intentionally embedded to compromise system integrity [4, 5].

Traditionally, subgraph matching in graphs is formulated as a *structural isomorphism* problem: determining whether a smaller query graph exactly matches part of a larger target graph in terms of node and edge connectivity. This problem is extensively studied in general graph theory, and classical approaches rely primarily on combinatorial search algorithms [6, 7, 8]. However, subgraph isomorphism is an NP-complete problem and thus often suffers from exponential computational complexity in worst-case scenarios. Recently, deep learning methods have emerged to mitigate this computational cost by embedding graphs into continuous latent spaces, significantly accelerating

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-3336

matching tasks [9, 10, 11]. Within the EDA domain, these techniques have been successfully adapted for transistor-level subcircuit identification [12].

However, structure-based matching methods encounter significant limitations in practical EDA tasks, as circuit topologies frequently undergo substantial transformations during logic synthesis and technology mapping. Equivalent logic functions can thus be realized through widely differing structural implementations, driven by design considerations such as timing performance, power consumption, or silicon area. Consequently, exact structural correspondence rarely persists throughout the design process, even when the underlying logic function remains unchanged. This inherent limitation severely restricts the utility of traditional structural matching techniques, particularly in applications requiring cross-stage queries—for example, identifying subgraphs from an abstract netlist (like an And-Inverter Graph, or AIG) within a synthesized, technology-mapped netlist.

Motivated by this critical gap, we introduce an approach explicitly designed to recognize logic functionality irrespective of structural differences. Specifically, our framework determines whether the logic represented by a query subgraph exists implicitly within a candidate graph, independent of structural transformations.

To formalize this, we propose two key concepts: (1) **functional subgraph**, representing the circuit logic containment relation independent of structure, and (2) **fuzzy boundary**, minimal graph regions encapsulating the query's logic despite unclear structural boundaries. Consequently, our methodology, termed ***functional subgraph matching***, addresses two sub-tasks: *1. Functional Subgraph Detection*: Determining whether the logic function of a query graph is implicitly contained within a candidate graph; *2. Fuzzy Boundary Identification*: Precisely locating the smallest possible region (the fuzzy boundary) in the candidate graph that encapsulates the query's logic.

To achieve these objectives, we propose a novel two-stage multi-modal framework. In the first stage, we train our model with intra-modal and inter-modal alignment across different graph modalities, enabling robust and cross-stage detection of functional subgraph. In the second stage, we fine-tune our model and formulate fuzzy boundary detection as a graph segmentation task, moving beyond prior approaches that treated boundary identification as an input-output classification problem [13, 14]. By leveraging information from nodes located within the true boundaries, our segmentation approach significantly enhances performance and continuity of fuzzy boundary prediction.

Our experiments demonstrate the effectiveness of the proposed framework. Evaluations conducted across several widely-used benchmarks, ITC99 [15], OpenABCD [16] and ForgeEDA [17], show that our approach significantly surpasses traditional structure-based methods. Specifically, our framework achieves an average accuracy of 93.8% for functional subgraph detection and attains a DICE score of 91.3% for fuzzy boundary detection tasks. In contrast, structure-based baseline methods typically exhibit near-random performance (accuracy close to 50%) and high variability in precision, recall, and F1-score, underscoring their limitations in capturing implicit functionality. To further validate our method's robustness and generalizability, we additionally propose three function-aware baseline variants by integrating different graph encoders into our framework.

In summary, the contributions of this work include:

- Introducing and formally defining the novel concept of functional subgraph matching, clearly distinguishing it from structural isomorphism and functional equivalence.
- Developing a two-stage multi-modal embedding framework, leveraging both intra-modal and inter-modal alignments to capture structure-agnostic and function-invariant graph representations. This allows effective functional subgraph detection across different modalities.
- Proposing an innovative approach for fuzzy boundary identification by formulating the task as a graph segmentation problem rather than a simple input-output classification problem, significantly enhancing boundary continuity and localization accuracy.

2 Preliminaries

2.1 Subgraph Isomorphism Matching

Subgraph isomorphism matching is a fundamental problem in graph theory with applications across bioinformatics [18], social network analysis [19], and knowledge graphs [20, 21]. We first recall the standard definition of subgraph isomorphism in Definition 1.

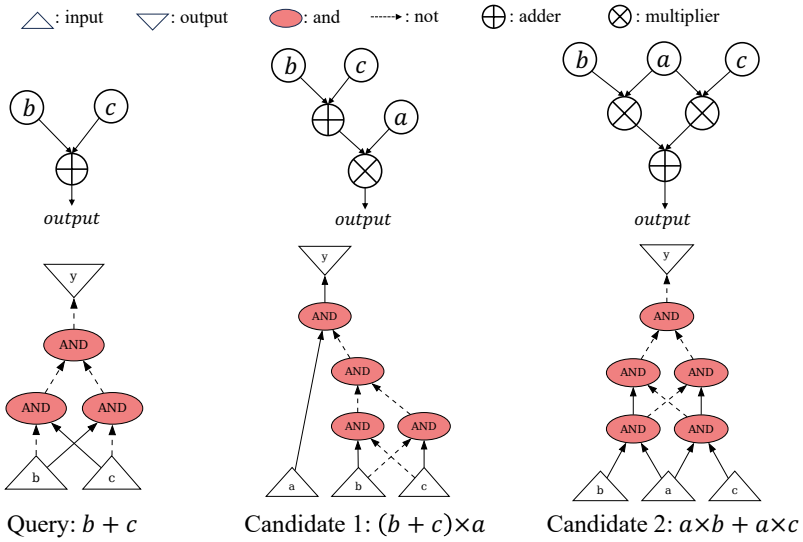

Figure 1: We present the query and candidate graphs. **Above**: 1-bit adder and multiplier. **Below**: AIG netlist. The query $b + c$ is explicitly contained within the candidate $(b + c) \times a$, making it straightforward to identify the exact subgraph in the candidate. In contrast, the query $b+c$ is implicitly contained within the candidate $a \times b + a \times c$, which implies no subgraph of $a \times b + a \times c$ has the same structure or function as the query graph.

Definition 1 (**Subgraph Isomorphism**). *A graph $\mathcal{Q}$ is an **isomorphic subgraph** of $\mathcal{G}$ if there exists a subgraph $\mathcal{G}'$ of $\mathcal{G}$ such that $\mathcal{Q}$ is isomorphic to $\mathcal{G}'$.*

Then, based on the definition of subgraph isomorphism, the subgraph isomorphism matching task is defined as follows: given a query graph $\mathcal{Q}$ and a target graph $\mathcal{G}$, determine if $\mathcal{Q}$ is isomorphic to a subgraph of $\mathcal{G}$. Classical approaches of subgraph isomorphism matching rely primarily on combinatorial search algorithms [7, 8, 6]. Its NP-complete nature, however, makes exact matching computationally intensive. More recently, graph-neural-network-based (GNN-based) methods have been introduced to learn compact graph embeddings that accelerate the matching process [9, 10, 11]. In the EDA domain, Li et al. [12] adapt the NeuroMatch architecture [10] to solve subcircuit isomorphism on transistor-level netlists.

However, in EDA flow, graphs often represent circuits or computations where structural modifications can preserve the underlying function. Standard subgraph isomorphism struggles with such cases. For instance, as illustrated in Figure 1, a model based on Definition 1 can identify that the structure representing $b + c$ is contained within $a \times (b + c)$, but it cannot identify the functional presence of $b + c$ within the structurally different but functionally related expression $a \times b + a \times c$.

2.2 Subgraph Equivalence

The limitation of structure-based subgraph matching motivates considering functional properties. Function-aware representation learning has emerged as a pivotal subfield in EDA. Many recent works emphasize functional equivalence, denoted $\mathcal{G}_1 \equiv_{func} \mathcal{G}_2$. DeepGate [22, 23, 24] and DeepCell [25] employ disentanglement to produce separate embeddings for functionality and structure, pretraining across various EDA benchmarks and predict functional similarity with a task head. PolarGate [26] enhances functional embeddings by integrating ambipolar device principles. FGNN [27, 28] applies contrastive learning to align circuit embeddings according to functional similarity.

While graph isomorphism requires structural identity, functional equivalence relates graphs based on their input-output behavior. Building on this, we can define a notion of subgraph relationship based on function, as shown in Definition 2.

Definition 2 (**Subgraph Equivalence**). *A graph $\mathcal{Q}$ is an **equivalent subgraph** of $\mathcal{G}$ if there exists a subgraph $\mathcal{G}'$ of $\mathcal{G}$ such that $\mathcal{Q} \equiv_{func} \mathcal{G}'$.*

https://doi.org/10.52202/085713-3336

This definition allows for functional matching within existing subgraphs. Some works adopt similar ideas for tasks such as arithmetic block identification [13, 29] and symbolic reasoning [14, 30], which aim to find a subgraph with specific functionality rather than structure. Compared to subgraph isomorphism, subgraph equivalence offers more flexibility against local structure modifications. However, Definition 2 still falls short for cases involving global restructuring. As shown in Figure 1, in the example $a \times b + a \times c$, no single subgraph is functionally equivalent to $b + c$. The function $b + c$ is **implicitly** present but not explicitly represented by a contiguous subgraph.

2.3 Functional Subgraph

To address the limitations of both Definition 1 and Definition 2, we introduce the concept of a functional subgraph, which aims to identify the implicit containment relation between graphs.

Definition 3 (Functional Subgraph). *A graph $\mathcal{Q}$ is a **functional subgraph** of $\mathcal{G}$, denoted $\mathcal{Q} \preccurlyeq \mathcal{G}$, if there exists a graph $\mathcal{G}'$ such that $\mathcal{G}' \equiv_{func} \mathcal{G}$ and $\mathcal{Q}$ is isomorphic to a subgraph of $\mathcal{G}'$.*

This definition captures the idea that the query's function is implicitly contained within the target's function, even if the target's structure has undergone functional transformations, and no exact subgraph isomorphic to the query graph can be found in the target graph. By this definition, we know that $b + c$ is a functional subgraph of $a \times b + a \times c$ since $a \times b + a \times c \equiv_{func} a \times (b + c)$ and $b + c$ is an isomorphic subgraph of $a \times (b + c)$. Furthermore, Definition 3 encompasses Definition 2, i.e., Definition 2 is a special case of Definition 3, as discussed in Proposition 1 (proof in Appendix A).

Proposition 1. *If a graph $\mathcal{Q}$ is an equivalent subgraph of $\mathcal{G}$, then $\mathcal{Q}$ is a functional subgraph of $\mathcal{G}$.*

Properties of Functional Subgraph In this paper, we assume that a graph obtained by removing some nodes and edges is not functionally equivalent to the original graph, i.e. $\forall g \neq \emptyset$, $G \setminus g \not\equiv_{\text{func}} G$. For example, we consider it illegal to directly connect two NOT gates. Therefore, such connections do not appear in our graph structures. In fact, EDA tools such as ABC [31] inherently enforce this constraint. According to Definition 3, functional subgraphs exhibit the following properties:

- ***Reflexivity***: For any graph $\mathcal{G}$, $\mathcal{G}$ is the functional subgraph of $\mathcal{G}$, i.e. $\forall \mathcal{G}, \mathcal{G} \preccurlyeq \mathcal{G}$.
- ***Functional Equivalence Preservation***: If $\mathcal{G}_1$ is a functional subgraph of $\mathcal{G}_2$, then:
  - (Left-hand Side) if $\mathcal{G}_1'$ is functionally equivalent to $\mathcal{G}_1$, then $\mathcal{G}_1'$ is a functional subgraph of $\mathcal{G}_2$, i.e. if $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_1' \equiv_{func} \mathcal{G}_1$, then $\mathcal{G}_1' \preccurlyeq \mathcal{G}_2$.
  - (Right-hand Side) if $\mathcal{G}_2'$ is functionally equivalent to $\mathcal{G}_2$, then $\mathcal{G}_1$ is a functional subgraph of $\mathcal{G}_2'$, i.e. if $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_2' \equiv_{func} \mathcal{G}_2$, then $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2'$.
- ***Transitivity***: If $\mathcal{G}_1$ is a functional subgraph of $\mathcal{G}_2$ and $\mathcal{G}_2$ is a functional subgraph of $\mathcal{G}_3$, then $\mathcal{G}_1$ is a functional subgraph of $\mathcal{G}_3$, i.e. if $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_2 \preccurlyeq \mathcal{G}_3$, then $\mathcal{G}_1 \preccurlyeq \mathcal{G}_3$.
- ***Anti-symmetry***: If $\mathcal{G}_1$ is a functional subgraph of $\mathcal{G}_2$, then $\mathcal{G}_2$ is a functional subgraph of $\mathcal{G}_1$ if and only if they are functionally equivalent, i.e. $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_2 \preccurlyeq \mathcal{G}_1$ if and only if $\mathcal{G}_1 \equiv_{func} \mathcal{G}_2$.

For detailed proofs of the above properties, please refer to Appendix A. It is worth noting that the subgraph equivalence defined in Definition 2 does **not** satisfy the *Transitivity* property. This highlights the improved completeness of the functional subgraph in Definition 3.

2.4 Task Definition

Based on Definition 3, we define our primary task:

Task #1: Functional Subgraph Detection. Given a query graph $\mathcal{Q}$ and a candidate graph $\mathcal{G}$, determine if $\mathcal{Q} \preccurlyeq \mathcal{G}$.

While functional subgraph detection is a decision problem (yes/no), it is often desirable to identify **which part** of the target graph $\mathcal{G}$ corresponds to the query function $\mathcal{Q}$. However, as shown in Figure 1, due to potential functional transformations, identifying an exact boundary in the original graph $\mathcal{G}$ that perfectly represents $\mathcal{Q}$ can be challenging or impossible. This leads to our second task, which aims to find the smallest region in $\mathcal{G}$ that encapsulates the function of $\mathcal{Q}$.

Definition 4 (Fuzzy Boundary). *Given a query graph $\mathcal{Q}$ and a candidate graph $\mathcal{G} = (V, E)$, a subgraph $\mathcal{G}^* = (V^*, E^*)$ of $\mathcal{G}$, where $V^* \subseteq V$ and $E^* = E \cap (V^* \times V^*)$, is a **fuzzy boundary** for $\mathcal{Q}$ in $\mathcal{G}$ if:*

1. *$\mathcal{Q} \preccurlyeq \mathcal{G}^*$*
2. *For any proper subgraph $\mathcal{H}$ of $\mathcal{G}^*$ (i.e., $\mathcal{H} \subset \mathcal{G}^*$ and $\mathcal{H} \neq \mathcal{G}^*$), $\mathcal{Q} \not\preccurlyeq \mathcal{H}$*

As illustrated in Figure 1, for $\mathcal{G}$ representing $a \times b + a \times c$ and $\mathcal{Q}$ representing $b + c$, the fuzzy boundary $\mathcal{G}^*$ would likely encompass the components corresponding to b, c, the two multiplications, and the addition, as this minimal collection is required to functionally contain $b+c$ via transformation. Based on Definition 4, we further define another task as:

Task #2: Fuzzy Boundary Identification. Given a query graph $\mathcal{Q}$ and a candidate graph $\mathcal{G}$ such that $\mathcal{Q} \preccurlyeq \mathcal{G}$, determine for each node in $\mathcal{G}$, whether it belongs to the fuzzy boundary $\mathcal{G}^*$ of $\mathcal{Q}$.

3 Method

3.1 Stage #1: Functional Subgraph Detection

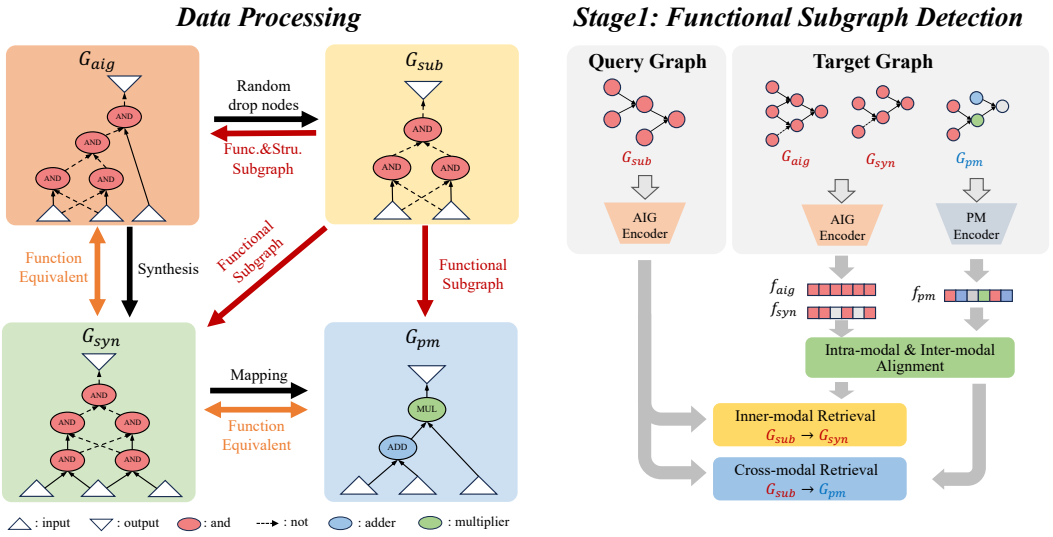

Figure 2: The pipeline of Stage #1. **Left**: Our data processing pipeline. For a given $\mathcal{G}_{aig}$, we first randomly extract a subgraph $\mathcal{G}_{sub}$. Then, we obtain $\mathcal{G}_{syn}$ and $\mathcal{G}_{pm}$ through synthesis and mapping, respectively. **Right**: Our training pipeline via intra-modal and inter-modal alignments for functional subgraph detection. We first encode the query and target graphs using their respective encoders. Next, we perform intra-modal and inter-modal alignment on the target graph to obtain function-invariant and structure-agnostic embeddings. These embeddings are then sent to a task head to determine whether the query graph is contained within the target graph.

Data Processing As illustrated in Figure 2, given an AIG netlist $\mathcal{G}_{aig}$, we first randomly drop nodes while ensuring legality, to obtain the subgraph $\mathcal{G}_{sub}$. Next, we use the ABC tool [31] to generate $\mathcal{G}_{syn}$ by randomly selecting a synthesis flow. Importantly, in this step we ensure that $\mathcal{G}_{syn}$ is not isomorphic to $\mathcal{G}_{aig}$. Finally, we apply the ABC tool again to map $\mathcal{G}_{syn}$ to $\mathcal{G}_{pm}$ using the Skywater Open Source PDK [32]. This data processing pipeline ensures that $\mathcal{G}_{aig}$ is equivalent to both $\mathcal{G}_{syn}$ and $\mathcal{G}_{pm}$. Since $\mathcal{G}_{sub}$ is an isomorphic subgraph of $\mathcal{G}_{aig}$, it follows from Definition 3 that $\mathcal{G}_{sub}$ is a functional subgraph of both $\mathcal{G}_{syn}$ and $\mathcal{G}_{pm}$. For negative pairs, following the approach in Li et al. [12], we randomly sample $\mathcal{G}_{aig}$, $\mathcal{G}_{syn}$, and $\mathcal{G}_{pm}$ from other pairs within the same batch. It is important to note that all circuits in this paper have multiple inputs and a single output. For more details, please refer to Section 4.1 and Appendix C.

https://doi.org/10.52202/085713-3336

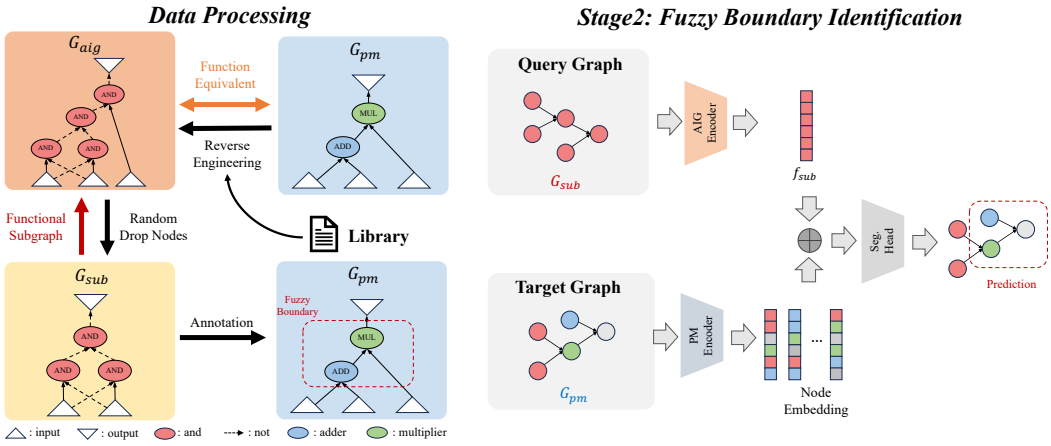

Figure 3: The pipeline of Stage #2. **Left**: Our data processing pipeline. For a given $\mathcal{G}_{pm}$, we replace each node in $\mathcal{G}_{pm}$ with the AIG implementation according to the functionality in the library. Then, we randomly sample a subgraph $\mathcal{G}_{sub}$ from $\mathcal{G}_{aig}$. Finally, we annotate the nodes in $\mathcal{G}_{pm}$ if one of the corresponding AIG nodes still exist in $\mathcal{G}_{sub}$. **Right**: Our training pipeline for fuzzy boundary identification via graph segmentation. Given the query graph $\mathcal{G}_{sub}$ and the target graph $\mathcal{G}_{pm}$, we first use Enc_{aig} to obtain the graph embedding of $\mathcal{G}_{sub}$ and Enc_{pm} to obtain the node embeddings of $\mathcal{G}_{pm}$. These embeddings are then concatenated and passed to a task head to determine whether a node in $\mathcal{G}_{pm}$ lies within the fuzzy boundary of $\mathcal{G}_{sub}$.

Retrieval In this paper, we adopt DeepGate2 [22] and DeepCell [25] as backbones for encoding AIG netlists and post-mapping netlists, respectively. Given a query graph $\mathcal{G}_{sub}$, along with positive candidates $\mathcal{G}^+_{aig}, \mathcal{G}^+_{syn}, \mathcal{G}^+_{pm}$ and negative candidates $\mathcal{G}^-_{aig}, \mathcal{G}^-_{syn}, \mathcal{G}^-_{pm}$, we first use the AIG encoder Enc_{aig} and the PM encoder Enc_{pm} for different modalities as follows:

$$f_{sub} = Enc_{aig}(\mathcal{G}_{sub}),\ f_{aig} = Enc_{aig}(\mathcal{G}_{aig})$$
$$f_{syn} = Enc_{aig}(\mathcal{G}_{syn}),\ f_{pm} = Enc_{pm}(\mathcal{G}_{pm})$$

Next, we concatenate the embeddings of the query graph and the candidate graphs and feed them into a classification head, a 3-layer MLP:

$$\hat{y}_{aig} = MLP([f_{sub}, f_{aig}]),\ \hat{y}_{syn} = MLP([f_{sub}, f_{syn}]),\ \hat{y}_{pm} = MLP([f_{sub}, f_{pm}])$$

Finally, we compute the binary cross-entropy (BCE) loss for each prediction:

$$L_{cls} = BCELoss(\hat{y}_{aig}, y_{aig}) + BCELoss(\hat{y}_{syn}, y_{syn}) + BCELoss(\hat{y}_{pm}, y_{pm})$$

Function-Invariant Alignment EDA flows such as synthesis and mapping modify the circuit structure while preserving functional equivalence. As defined in Definition 3, the functional subgraph relation focuses on the functionality of the candidate circuits rather than structure, as they can be transformed into an equivalent circuit with any structure. Furthermore, the *Functional Equivalence Preservation* property in Section 2.3 imply that, if the subgraph relation $Q \preccurlyeq G$ hold, then if we replace Q or G with another functional equivalent graph, the subgraph relation continues to hold. This invariance is the key insight for our alignment: the embeddings of functionally equivalent graphs should be aligned, regardless of their structural variations.

Therefore, learning function-invariant embeddings for equivalent circuits across different stages is crucial for functional subgraph detection. While $\mathcal{G}_{aig}$ and $\mathcal{G}_{syn}$ share the same gate types, $\mathcal{G}_{aig}$ and $\mathcal{G}_{pm}$ differ significantly in modality, i.e., the gate types in $\mathcal{G}_{pm}$ are substantially dissimilar to those in $\mathcal{G}_{aig}$. Therefore, we employ both intra-modal and inter-modal alignment techniques to acquire function-invariant and structure-agnostic embeddings with the InfoNCE loss [33]. We select $\mathcal{G}_{aig}$ as the anchor and compute the intra-modal and inter-modal losses as follows:

$$L_{intra} = InfoNCE(f^+_{aig}, f^+_{syn}, f^-_{syn})$$
$$L_{inter} = InfoNCE(f^+_{aig}, f^+_{pm}, f^-_{pm})$$

Finally, we summarize the losses for stage #1 as:

$$L_{stage_1} = L_{cls} + L_{intra} + L_{inter}$$

3.2 Stage #2: Fuzzy Boundary Identification

Data Processing As illustrated in Figure 3, given a post-mapping netlist $\mathcal{G}_{pm}$, we replace the cells in $\mathcal{G}_{pm}$ with the corresponding AIGs from the library to acquire the netlist $\mathcal{G}_{aig}$. This process yields a mapping function ϕ that associates the node indices of $\mathcal{G}_{aig}$ with those of $\mathcal{G}_{pm}$. Next, we randomly drop nodes to obtain $\mathcal{G}_{sub}$, which serves as the functional subgraph of both $\mathcal{G}_{pm}$ and $\mathcal{G}_{aig}$. Using the subgraph $\mathcal{G}_{sub}$, we annotate the nodes in $\mathcal{G}_{pm}$ by mapping the node indices of $\mathcal{G}_{sub}$ to those of $\mathcal{G}_{pm}$ through the function ϕ. Specifically, for each node in $\mathcal{G}_{sub}$, if it maps to a node i in $\mathcal{G}_{pm}$, we annotate node i as 1; otherwise, we annotate it as 0. This annotation process strictly follows the fuzzy boundary definition in Definition 4.

Cross-modal Retrieval Given a query graph $\mathcal{G}_{sub}$ and a target graph $\mathcal{G}_{pm} = (V_{pm}, E_{pm})$, we first compute the embedding of $\mathcal{G}_{sub}$ and the node embeddings of $\mathcal{G}_{pm}$:

$$f_{sub} = Enc_{aig}(\mathcal{G}_{sub}),\ f_{pm}^1, f_{pm}^2, \ldots, f_{pm}^{|V_{pm}|} = Enc_{pm}(\mathcal{G}_{pm})$$

Next, we use f_{sub} as the query embedding and concatenate it with the node embeddings from $\mathcal{G}_{pm}$. These concatenated embeddings are then fed into a 3-layer MLP for node classification: $\hat{y}_i = MLP([f_{sub}, f_{pm}^i])$. While previous works [13, 29] treat this task as an input-output classification problem, we frame it as a graph segmentation task. This approach arises from the observation that nodes near the input-output nodes contribute to identifying fuzzy boundaries and thus should not be simply labeled as zero. During training, we optimize the model using cross-entropy loss:

$$L_{stage_2} = -\sum_i [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{1}$$

4 Experiment

4.1 Experimental Setup

We evaluate our method on three AIG datasets: ITC99 [15], OpenABCD [16], and ForgeEDA [17]. Each metric in Tables 1 and 2 is reported as the *mean* $\pm$ *standard deviation* over three independent runs. For data processing, we begin by randomly sampling k-hop subgraphs (with k ranging from 8 to 12) to partition large circuits into smaller circuits. Next, we randomly sample subgraphs from these smaller circuits. For logic synthesis, we use the ABC tool [31] with a randomly selected flow from *src_rw*, *src_rs*, *src_rws*, *resyn2rs*, and *compress2rs*. We then apply the VF2 algorithm [6] to verify that the synthesis process has modified the circuit structure. If no modification is detected, we repeat this step until we obtain a circuit with a different structure. For technology mapping, we invoke ABC with the Skywater Open Source PDK [32]. For additional details on the environment, evaluation metrics, and dataset statistics, please refer to Appendix C.

Table 1: Result of Functional Subgraph Detection(%).

Dataset	Method	$\mathcal{G}_{sub} \to \mathcal{G}_{syn}$				$\mathcal{G}_{sub} \to \mathcal{G}_{pm}$			
		Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
ITC99	NeuroMatch	$49.8_{\pm 0.3}$	$16.7_{\pm 23.6}$	$33.3_{\pm 47.1}$	$22.2_{\pm 31.4}$	$49.8_{\pm 0.2}$	$16.7_{\pm 23.6}$	$50.0_{\pm 50.0}$	$33.4_{\pm 33.4}$
	HGCN	$44.5_{\pm 7.7}$	$35.0_{\pm 21.2}$	$67.3_{\pm 46.3}$	$45.3_{\pm 30.2}$	$49.5_{\pm 0.8}$	$35.7_{\pm 20.2}$	$66.8_{\pm 47.0}$	$44.7_{\pm 31.2}$
	Gamora	$50.6_{\pm 12.8}$	$21.1_{\pm 27.7}$	$33.0_{\pm 46.0}$	$25.4_{\pm 34.8}$	$51.7_{\pm 4.4}$	$34.0_{\pm 24.2}$	$51.2_{\pm 40.9}$	$40.2_{\pm 29.6}$
	ABGNN	$56.4_{\pm 9.1}$	$20.8_{\pm 29.4}$	$32.7_{\pm 46.3}$	$25.4_{\pm 35.9}$	$54.1_{\pm 5.8}$	$19.0_{\pm 26.9}$	$33.3_{\pm 47.1}$	$24.2_{\pm 34.2}$
	Ours	$\mathbf{95.3_{\pm 0.1}}$	$\mathbf{94.4_{\pm 0.2}}$	$\mathbf{96.3_{\pm 0.1}}$	$\mathbf{95.4_{\pm 0.0}}$	$\mathbf{93.1_{\pm 0.3}}$	$\mathbf{92.3_{\pm 0.3}}$	$\mathbf{94.2_{\pm 0.9}}$	$\mathbf{93.2_{\pm 0.4}}$
OpenABCD	NeuroMatch	$44.2_{\pm 9.8}$	$17.3_{\pm 23.9}$	$33.4_{\pm 47.1}$	$22.7_{\pm 31.8}$	$44.9_{\pm 8.4}$	$17.0_{\pm 23.9}$	$33.4_{\pm 47.1}$	$22.5_{\pm 31.7}$
	HGCN	$52.5_{\pm 3.6}$	$18.0_{\pm 25.5}$	$32.5_{\pm 46.0}$	$23.2_{\pm 32.8}$	$50.0_{\pm 0.0}$	$20.4_{\pm 21.4}$	$33.0_{\pm 46.7}$	$22.2_{\pm 31.3}$
	Gamora	$50.8_{\pm 1.1}$	$33.7_{\pm 23.9}$	$66.6_{\pm 47.1}$	$44.8_{\pm 31.7}$	$49.8_{\pm 0.3}$	$33.2_{\pm 23.5}$	$62.1_{\pm 44.3}$	$43.3_{\pm 30.6}$
	ABGNN	$34.1_{\pm 5.4}$	$5.2_{\pm 3.9}$	$2.6_{\pm 2.6}$	$3.4_{\pm 3.2}$	$41.3_{\pm 4.0}$	$9.7_{\pm 7.6}$	$3.5_{\pm 3.2}$	$5.1_{\pm 4.5}$
	Ours	$\mathbf{92.3_{\pm 0.2}}$	$\mathbf{93.7_{\pm 0.2}}$	$\mathbf{90.6_{\pm 0.4}}$	$\mathbf{92.1_{\pm 0.2}}$	$\mathbf{90.8_{\pm 0.4}}$	$\mathbf{92.4_{\pm 0.4}}$	$\mathbf{88.9_{\pm 0.9}}$	$\mathbf{90.6_{\pm 0.5}}$
ForgeEDA	NeuroMatch	$50.0_{\pm 0.0}$	$16.7_{\pm 23.6}$	$33.3_{\pm 47.1}$	$22.2_{\pm 31.4}$	$50.0_{\pm 0.0}$	$16.7_{\pm 23.6}$	$33.3_{\pm 47.1}$	$22.2_{\pm 31.4}$
	HGCN	$44.0_{\pm 8.5}$	$18.2_{\pm 22.6}$	$33.9_{\pm 46.7}$	$23.1_{\pm 30.8}$	$48.8_{\pm 1.6}$	$18.8_{\pm 22.2}$	$33.5_{\pm 47.0}$	$22.5_{\pm 31.2}$
	Gamora	$40.6_{\pm 6.3}$	$2.4_{\pm 1.6}$	$0.7_{\pm 0.8}$	$1.0_{\pm 1.1}$	$48.2_{\pm 1.5}$	$51.0_{\pm 8.2}$	$31.0_{\pm 31.6}$	$28.5_{\pm 22.9}$
	ABGNN	$52.3_{\pm 3.3}$	$34.6_{\pm 24.5}$	$66.6_{\pm 47.1}$	$45.5_{\pm 32.2}$	$52.0_{\pm 2.9}$	$34.4_{\pm 24.4}$	$66.6_{\pm 47.1}$	$45.4_{\pm 32.1}$
	Ours	$\mathbf{96.0_{\pm 0.1}}$	$\mathbf{96.8_{\pm 0.4}}$	$\mathbf{95.2_{\pm 0.5}}$	$\mathbf{96.0_{\pm 0.1}}$	$\mathbf{95.3_{\pm 0.0}}$	$\mathbf{95.9_{\pm 0.5}}$	$\mathbf{94.7_{\pm 0.5}}$	$\mathbf{95.3_{\pm 0.0}}$

https://doi.org/10.52202/085713-3336

4.2 Stage #1: Functional Subgraph Detection

We evaluate the performance of our proposed method on three datasets: ITC99, OpenABCD, and ForgeEDA. Our method is compared against several state-of-the-art models, including NeuroMatch [10] and HGCN [12], which are designed for isomorphism subgraph matching in general domain and EDA domain respectively, and Gamora [14] and ABGNN [13], which are designed for reasoning in EDA domain, i.e. for equivalent subgraph matching. Since Gamora and ABGNN focus on boundary detection instead of subgraph matching, we integrate them into the NeuroMatch framework for Stage #1. Further integration of Gamora and ABGNN with our method is discussed in Appendix B. The evaluation metrics include accuracy, precision, recall, and F1-score, computed for two tasks: $\mathcal{G}_{sub}$ to $\mathcal{G}_{syn}$ and $\mathcal{G}_{sub}$ to $\mathcal{G}_{pm}$.

As shown in Table 1, the results on the ITC99, OpenABCD, and ForgeEDA datasets demonstrate that our method significantly outperforms all baseline models. Specifically, for the $\mathcal{G}_{sub} \rightarrow \mathcal{G}_{syn}$ task, our model achieves an average accuracy of $\mathbf{94.5}\%$, precision of $\mathbf{95.0}\%$, recall of $\mathbf{94.0}\%$, and F1-score of $\mathbf{94.5}\%$, surpassing all other methods by a large margin. Similarly, for the $\mathcal{G}_{sub} \rightarrow \mathcal{G}_{pm}$ task, our method also shows superior performance with an accuracy of $\mathbf{93.1}\%$, precision of $\mathbf{93.5}\%$, recall of $\mathbf{92.6}\%$, and F1-score of $\mathbf{93.0}\%$. In contrast, structure-based methods show an accuracy close to 50% and large standard errors in precision, recall, and F1-score. Such unreliable performance typically arises because these methods indiscriminately predict all pairs as either entirely positive or negative, highlighting their limitations in functional subgraph detection.

4.3 Stage #2: Fuzzy Boundary Identification

Table 2: Result of Fuzzy Boundary Identification(%).

Method	ITC99		OpenABCD		ForgeEDA	
	IoU	DICE	IoU	DICE	IoU	DICE
NeuroMatch	$44.2_{\pm 0.0}$	$61.3_{\pm 0.0}$	$41.2_{\pm 0.0}$	$58.3_{\pm 0.0}$	$42.0_{\pm 0.0}$	$59.1_{\pm 0.0}$
HGCN	$44.1_{\pm 0.0}$	$61.2_{\pm 0.0}$	$41.2_{\pm 0.0}$	$58.3_{\pm 0.0}$	$42.0_{\pm 0.0}$	$59.2_{\pm 0.0}$
Gamora	$39.1_{\pm 2.8}$	$56.2_{\pm 2.9}$	$44.2_{\pm 1.2}$	$61.3_{\pm 1.1}$	$39.5_{\pm 0.6}$	$56.6_{\pm 0.6}$
ABGNN	$26.7_{\pm 6.2}$	$41.7_{\pm 7.5}$	$37.5_{\pm 0.8}$	$54.5_{\pm 0.8}$	$31.9_{\pm 2.6}$	$48.2_{\pm 3.0}$
Ours	$\mathbf{83.0}_{\pm \mathbf{1.4}}$	$\mathbf{90.7}_{\pm \mathbf{0.9}}$	$\mathbf{85.2}_{\pm \mathbf{0.9}}$	$\mathbf{92.0}_{\pm \mathbf{0.5}}$	$\mathbf{83.8}_{\pm \mathbf{0.8}}$	$\mathbf{91.2}_{\pm \mathbf{0.4}}$

In this stage, we treat $\mathcal{G}_{sub}$ as the query and aim to locate its fuzzy boundary within the post-mapping netlist $\mathcal{G}_{pm}$. Since Gamora and ABGNN are designed for the detection of the input-output boundary, we first apply each to identify the input and output nodes in $\mathcal{G}_{pm}$. We then perform a BFS between inputs and outputs to recover the corresponding fuzzy boundary, and evaluate the result using Intersection-over-Union (IoU) and DICE score.

Table 2 reports results on ITC99, OpenABCD, and ForgeEDA, demonstrating that our model substantially outperforms all baselines. Specifically, we achieve an average IoU of $\mathbf{84.0}\%$ and a Dice score of $\mathbf{91.3}\%$, significantly outperforming all other methods. Structure-based methods (e.g., NeuroMatch and HGCN) fail to capture functional boundaries and often generate trivial solutions (predicting all nodes as boundary nodes), yielding low variance but poor performance. Although Gamora and ABGNN can detect clear block boundaries for specific arithmetic modules, they struggle with the variable, function-driven fuzzy boundaries required here, resulting in significantly lower performance. Further integration of Gamora and ABGNN within our framework is detailed in Appendix B.

4.4 Ablation Study

We perform ablation study on ITC99 dataset and compare the performance of the ablation settings with our proposed method to evaluate the contribution of various components in our method.

Stage #1 without alignment achieves accuracy and F1-scores of 94.6% and 94.6% on $\mathcal{G}_{sub} \rightarrow \mathcal{G}_{syn}$ task, which are lower than our method's 95.3% and 95.4%. Our model also improves accuracy and F1-score by 1.7% on $\mathcal{G}_{sub} \rightarrow \mathcal{G}_{pm}$ task. These results demonstrate the importance of function-invariant alignment, particularly inter-modal alignment, i.e. aligning $\mathcal{G}_{pm}$ and $\mathcal{G}_{aig}$.

Table 3: Ablation Study on ITC99 Dataset(%).

Setting	Stage #1				Stage #2	
	$\mathcal{G}_{sub} \rightarrow \mathcal{G}_{syn}$		$\mathcal{G}_{sub} \rightarrow \mathcal{G}_{pm}$		$\mathcal{G}_{sub} \rightarrow \mathcal{G}_{pm}$	
	Accuracy	F1-score	Accuracy	F1-score	IoU	DICE
Stage #1 *wo.* alignment	94.6	94.6	91.4	91.5	-	-
Stage #2 *wo.* stage #1	-	-	-	-	76.3	86.5
Stage #2 *wo.* seg.	-	-	-	-	29.6	45.7
Ours	**95.3**	**95.4**	**93.1**	**93.2**	**83.0**	**90.7**

Stage #2 without Stage #1 shows a performance drop, with IoU and DICE scores of 76.3% and 86.5%, compared to our method's improved values of 83.0% and 90.7%. This highlights the crucial role of pretraining knowledge in Stage #1.

Stage #2 without segmentation also shows a significant drop in performance, with IoU and DICE values of 29.6% and 45.7%, compared to our method's improved 83.0% and 90.7%. These results suggest that directly predicting the input-output nodes of the fuzzy boundary is challenging, as it varies with different functional transformations and omits the information of nodes in fuzzy boundary.

5 Limitations

While our proposed framework demonstrates strong performance and significant improvements over existing structural approaches, several limitations remain and should be addressed in future research:

Scalability to Large-scale Circuits: Currently, our method has primarily been evaluated on moderately-sized circuits due to computational resource constraints. Real-world EDA applications often involve extremely large netlists with millions of nodes. Scaling our detection and segmentation approaches to handle such large-scale graphs efficiently is non-trivial. Future research could investigate more computationally efficient embedding methods, hierarchical segmentation approaches, or incremental graph processing techniques to enhance scalability.

Multiple and Overlapping Fuzzy Boundaries: Our fuzzy boundary identification method presently assumes a single, minimal enclosing region within the target graph. In practical scenarios, multiple occurrences or overlapping functional subgraphs might exist within a single large circuit, complicating boundary identification tasks. Extending our methodology to effectively handle multiple or overlapping fuzzy boundaries within the same circuit remains an open and challenging direction for further investigation.

Single-output Circuit Assumption: The current approach assumes single-output logic circuits. In real-world scenarios, however, most circuits possess multiple outputs and complex internal functional dependencies. The direct applicability of our method to multi-output circuits, particularly when outputs share significant internal logic, remains unexplored. Generalizing the definitions and embedding strategies to model multi-output scenarios could further enhance practical relevance.

Non-trivial Function Assumption: In this paper, we assume that a graph obtained by removing some nodes and edges is not functionally equivalent to the original graph, i.e. $\forall g \neq \emptyset$, $G \setminus g \not\equiv_{\text{func}} G$. While EDA tools inherently enforce this constraint, it may limit the generalizability of the functional subgraph in other domains.

By systematically addressing these limitations, subsequent research can extend our approach to broader, more realistic settings, thereby increasing its practical utility in EDA domain and beyond.

6 Conclusion

In this paper, we introduce the concept of *functional subgraph matching*, a method to identify implicit logic functions within larger circuits, despite structural variations. We propose a two-stage framework: first, we train models across different modalities with alignment to detect functional subgraphs; second, we fine-tune our model and treat fuzzy boundary identification as a graph segmentation task for precise localization of fuzzy boundary. Evaluations on benchmarks (ITC99, OpenABCD, ForgeEDA) show that our approach outperforms structure-based methods, achieving 93.8% accuracy in functional subgraph detection and a 91.3% DICE score for fuzzy boundary detection.

 https://doi.org/10.52202/085713-3336

Broader Impact Our method contributes to the advancement of deep learning, particularly in graph-based functional relationship analysis. By improving the detection of functional relationships in complex systems, it has the potential to impact a wide range of applications, from circuit design to other domains that rely on graph functionality, e.g. molecular and protein graphs.

Acknowledgment This work was supported in part by the Hong Kong Research Grants Council (RGC) under Grant No. 14212422, 14202824, and C6003-24Y.

References

[1] Alireza Mahzoon, Daniel Große, and Rolf Drechsler. Polycleaner: clean your polynomials before backward rewriting to verify million-gate multipliers. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[2] Alireza Mahzoon, Daniel Große, and Rolf Drechsler. Revsca: Using reverse engineering to bring light into backward rewriting for big and dirty multipliers. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[3] Xing Wei, Yi Diao, Tak-Kei Lam, and Yu-Liang Wu. A universal macro block mapping scheme for arithmetic circuits. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1629–1634. IEEE, 2015.

[4] Travis Meade, Shaojie Zhang, Yier Jin, Zheng Zhao, and David Pan. Gate-level netlist reverse engineering tool set for functionality recovery and malicious logic detection. In *International Symposium for Testing and Failure Analysis*, volume 81368, pages 342–346. ASM International, 2016.

[5] Haocheng Li, Satwik Patnaik, Abhrajit Sengupta, Haoyu Yang, Johann Knechtel, Bei Yu, Evangeline FY Young, and Ozgur Sinanoglu. Attacking split manufacturing from a deep learning perspective. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[6] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1367–1372, 2004.

[7] Julian R Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23 (1):31–42, 1976.

[8] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, Mario Vento, et al. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pages 149–159. Citeseer, 2001.

[9] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 384–392, 2019.

[10] Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, Jure Leskovec, et al. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*, 2020.

[11] Rex Ying, Tianyu Fu, Andrew Wang, Jiaxuan You, Yu Wang, and Jure Leskovec. Representation learning for frequent subgraph mining. *arXiv preprint arXiv:2402.14367*, 2024.

[12] Bohao Li, Shizhang Wang, Tinghuan Chen, Qi Sun, and Cheng Zhuo. Efficient subgraph matching framework for fast subcircuit identification. In *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*, pages 1–7, 2024.

[13] Ziyi Wang, Zhuolun He, Chen Bai, Haoyu Yang, and Bei Yu. Efficient arithmetic block identification with graph learning and network-flow. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(8):2591–2603, 2022.

[14] Nan Wu, Yingjie Li, Cong Hao, Steve Dai, Cunxi Yu, and Yuan Xie. Gamora: Graph learning based symbolic reasoning for large-scale boolean networks. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[15] Scott Davidson. Characteristics of the itc'99 benchmark circuits. In *ITSW*, 1999.

[16] Animesh Basak Chowdhury, Benjamin Tan, Ramesh Karri, and Siddharth Garg. Openabc-d: A large-scale dataset for machine learning guided integrated circuit synthesis. *arXiv preprint arXiv:2110.11292*, 2021.

[17] Zhengyuan Shi, Zeju Li, Chengyu Ma, Yunhao Zhou, Ziyang Zheng, Jiawei Liu, Hongyang Pan, Lingfeng Zhou, Kezhi Li, Jiaying Zhu, et al. Forgeeda: A comprehensive multimodal dataset for advancing eda. *arXiv preprint arXiv:2505.02016*, 2025.

[18] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14:1–13, 2013.

[19] Wenfei Fan. Graph pattern matching revised for social network analysis. In *Proceedings of the 15th international conference on database theory*, pages 8–21, 2012.

[20] Jinha Kim, Hyungyu Shin, Wook-Shin Han, Sungpack Hong, and Hassan Chafi. Taming subgraph isomorphism for rdf query processing. *arXiv preprint arXiv:1506.01973*, 2015.

[21] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45, 2009.

[22] Zhengyuan Shi, Hongyang Pan, Sadaf Khan, Min Li, Yi Liu, Junhua Huang, Hui-Ling Zhen, Mingxuan Yuan, Zhufei Chu, and Qiang Xu. Deepgate2: Functionality-aware circuit representation learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2023.

[23] Zhengyuan Shi, Ziyang Zheng, Sadaf Khan, Jianyuan Zhong, Min Li, and Qiang Xu. Deepgate3: Towards scalable circuit representation learning. *arXiv preprint arXiv:2407.11095*, 2024.

[24] Ziyang Zheng, Shan Huang, Jianyuan Zhong, Zhengyuan Shi, Guohao Dai, Ningyi Xu, and Qiang Xu. Deepgate4: Efficient and effective representation learning for circuit design at scale. *arXiv preprint arXiv:2502.01681*, 2025.

[25] Zhengyuan Shi, Chengyu Ma, Ziyang Zheng, Lingfeng Zhou, Hongyang Pan, Wentao Jiang, Fan Yang, Xiaoyan Yang, Zhufei Chu, and Qiang Xu. Deepcell: Multiview representation learning for post-mapping netlists. *arXiv preprint arXiv:2502.06816*, 2025.

[26] Jiawei Liu, Jianwang Zhai, Mingyu Zhao, Zhe Lin, Bei Yu, and Chuan Shi. Polargate: Breaking the functionality representation bottleneck of and-inverter graph neural network. In *2024 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024.

[27] Ziyi Wang, Chen Bai, Zhuolun He, Guangliang Zhang, Qiang Xu, Tsung-Yi Ho, Bei Yu, and Yu Huang. Functionality matters in netlist representation learning. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 61–66, 2022.

[28] Ziyi Wang, Chen Bai, Zhuolun He, Guangliang Zhang, Qiang Xu, Tsung-Yi Ho, Yu Huang, and Bei Yu. Fgnn2: A powerful pre-training framework for learning the logic functionality of circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[29] Zhuolun He, Ziyi Wang, Chen Bai, Haoyu Yang, and Bei Yu. Graph learning-based arithmetic block identification. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2021.

[30] Chenhui Deng, Zichao Yue, Cunxi Yu, Gokce Sarar, Ryan Carey, Rajeev Jain, and Zhiru Zhang. Less is more: Hop-wise graph attention for scalable and generalizable learning on circuits. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024.

 https://doi.org/10.52202/085713-3336

[31] Robert Brayton and Alan Mishchenko. Abc: An academic industrial-strength verification tool. In *CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings 22*, pages 24–40. Springer, 2010.

[32] Google. Skywater open source pdk. URL `https://github.com/google/skywater-pdk.git`. 2020.

[33] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[34] Vincenzo Carletti, Pasquale Foggia, Alessia Saggese, and Mario Vento. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with vf3. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):804–818, 2017.

A Proofs of the Proposed Properties

In this section, we use $G_1 \cong G_2$ to denote that G_1 is isomorphic to G_2. Also, we use $G_1 \equiv_{\text{func}} G_2$ to denote that G_1 is functional equivalent to G_2.

Proposition 2. *If a graph $\mathcal{Q}$ is an equivalent subgraph of $\mathcal{G}$, then $\mathcal{Q}$ is a functional subgraph of $\mathcal{G}$.*

Proof. According to the Definition 2, there exists a subgraph $\mathcal{G}'$ of $\mathcal{G}$ such that $\mathcal{Q} \equiv_{func} \mathcal{G}'$. By replacing $\mathcal{G}'$ with $\mathcal{Q}$, we get $\bar{\mathcal{G}} = \mathcal{G} \setminus \mathcal{G}' \cup \mathcal{Q}$ which is equivalent to $\mathcal{G}$ and a subgraph of $\mathcal{G}$ is isomorphic to $\mathcal{Q}$. By the Definition 3, $\mathcal{Q}$ is a functional subgraph of $\mathcal{G}$. □

Proposition 3 (Reflexivity)**.** $\forall \mathcal{G}, \mathcal{G} \preccurlyeq \mathcal{G}$.

Proof. $\mathcal{G}$ is a subgraph of itself, and $\mathcal{G} \equiv_{\text{func}} \mathcal{G}$. By the definition of functional subgraph, it follows that $\mathcal{G} \preccurlyeq \mathcal{G}$. □

Proposition 4 (Functional Equivalence Preservation)**.** *If $\mathcal{G}_1$ is a functional subgraph of $\mathcal{G}_2$, then:*

- *(Left-hand Side) if $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_1' \equiv_{func} \mathcal{G}_1$, then $\mathcal{G}_1' \preccurlyeq \mathcal{G}_2$.*
- *(Right-hand Side) if $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_2' \equiv_{func} \mathcal{G}_2$, then $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2'$.*

Proof. ***(Right-hand Side)*** According to the Definition 3, if $\mathcal{G}_1$ is a functional subgraph of $\mathcal{G}_2$, then there exist $\mathcal{G}_2^*$ that $\mathcal{G}_2^* \equiv_{\text{func}} \mathcal{G}_2$ and $\mathcal{G}_1$ is an isomorphic subgraph of $\mathcal{G}_2^*$. Since $\mathcal{G}_2' \equiv_{\text{func}} \mathcal{G}_2$ and $\mathcal{G}_2^* \equiv_{\text{func}} \mathcal{G}_2$, then $\mathcal{G}_2^*$. Since $\mathcal{G}_2' \equiv_{\text{func}} \mathcal{G}_2'$. By the Definition 3, $\mathcal{G}_1$ is a functional subgraph of $\mathcal{G}_2'$.

(Left-hand Side) By Definition 3, there exists a graph $\mathcal{G}_2' \equiv_{\text{func}} \mathcal{G}_2$, such that

$$\mathcal{G}_1 \cong \bar{\mathcal{G}}_2, \tag{2}$$

where $\bar{\mathcal{G}}_2$ is a subgraph of $\mathcal{G}_2'$. Since $\mathcal{G}_1 \cong \bar{\mathcal{G}}_2$, it follows that

$$\mathcal{G}_1 \equiv_{\text{func}} \bar{\mathcal{G}}_2. \tag{3}$$

By the transitivity of functional equivalence, we then have

$$\mathcal{G}_1 \equiv_{\text{func}} \bar{\mathcal{G}}_2 \equiv_{\text{func}} \mathcal{G}_1'. \tag{4}$$

Thus, by replacing $\bar{\mathcal{G}}_2$ in $\mathcal{G}_2'$ with $\mathcal{G}_1'$, we obtain a new graph

$$\mathcal{G}_2'' = (\mathcal{G}_2' \setminus \bar{\mathcal{G}}_2) \cup \mathcal{G}_1', \tag{5}$$

which satisfies

$$\mathcal{G}_2'' \equiv_{\text{func}} \mathcal{G}_2. \tag{6}$$

From the definition of functional equivalence, we know that $\mathcal{G}_2'' \equiv_{\text{func}} \mathcal{G}_2$ and that $\mathcal{G}_1'$ is a subgraph of $\mathcal{G}_2''$. Therefore, it follows that

$$\mathcal{G}_1' \preccurlyeq \mathcal{G}_2. \tag{7}$$

□

Proposition 5 (Transitivity)**.** *If $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_2 \preccurlyeq \mathcal{G}_3$, then $\mathcal{G}_1 \preccurlyeq \mathcal{G}_3$.*

Proof. By definition, there exists a graph $\mathcal{G}_2' \equiv_{\text{func}} \mathcal{G}_2$, such that

$$\mathcal{G}_1 \cong \bar{\mathcal{G}}_1, \text{and } \bar{\mathcal{G}}_1 \text{is a subgraph of } \mathcal{G}_2'. \tag{8}$$

Since $\mathcal{G}_2' \equiv_{\text{func}} \mathcal{G}_2$, by Proposition 4, it follows that $\mathcal{G}_2' \preccurlyeq \mathcal{G}_3$.

Therefore, there exists a graph $\mathcal{G}_3' \equiv_{\text{func}} \mathcal{G}_3$, and $\mathcal{G}_2'$ is a subgraph of $\mathcal{G}_3'$. Since $\bar{\mathcal{G}}_1$ is a subgraph of $\mathcal{G}_2'$ and $\mathcal{G}_2'$ is a subgraph of $\mathcal{G}_3'$, it follows that $\bar{\mathcal{G}}_1$ is a subgraph of $\mathcal{G}_3'$.

Since $\mathcal{G}_1 \cong \bar{\mathcal{G}}_1$, $\mathcal{G}_3' \equiv_{\text{func}} \mathcal{G}_3$ and $\bar{\mathcal{G}}_1$ is a subgraph of $\mathcal{G}_3'$, by the definition of functional subgraph, we conclude that

$$\mathcal{G}_1 \preccurlyeq \mathcal{G}_3. \tag{9}$$

□

Proposition 6 (Anti-symmetry)**.** *$\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_2 \preccurlyeq \mathcal{G}_1$ if and only if $\mathcal{G}_1 \equiv_{\text{func}} \mathcal{G}_2$.*

Proof. ($\Rightarrow$) Since $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$, we have

$$\mathcal{G}_1 \cong \mathcal{G}_2' \setminus g, \quad \text{and} \quad \mathcal{G}_2 \equiv_{\text{func}} \mathcal{G}_2'. \tag{10}$$

Since $\mathcal{G}_2 \preccurlyeq \mathcal{G}_1$, we have $\mathcal{G}_2' \preccurlyeq \mathcal{G}_2' \setminus g$. By the definition of functional subgraphs, there exists a graph $\mathcal{G}_3$ such that $\mathcal{G}_3 \equiv_{\text{func}} \mathcal{G}_2' \setminus g$ and $\mathcal{G}_2'$ is a subgraph of $\mathcal{G}_3$. This implies that $\mathcal{G}_2' \cong \mathcal{G}_3 \setminus g'$, so we also have

$$\mathcal{G}_2' \equiv_{\text{func}} \mathcal{G}_3 \setminus g'. \tag{11}$$

Since $\mathcal{G}_3 \equiv_{\text{func}} \mathcal{G}_2' \setminus g$, it follows that

$$\mathcal{G}_3 \cup g \equiv_{\text{func}} \mathcal{G}_2'. \tag{12}$$

Thus, we have

$$\mathcal{G}_3 \cup g \equiv_{\text{func}} \mathcal{G}_2' \equiv_{\text{func}} \mathcal{G}_3 \setminus g'. \tag{13}$$

Note that in Section 2.3, we assume that a graph obtained by removing some nodes and edges is not functionally equivalent to the original graph, i.e., $\forall g \neq \emptyset, G \setminus g \not\equiv_{\text{func}} G$. Therefore, we must have $g = g' = \emptyset$, which implies

$$\mathcal{G}_1 \cong \mathcal{G}_2' \setminus g \cong \mathcal{G}_2', \quad \text{and} \quad \mathcal{G}_2 \equiv_{\text{func}} \mathcal{G}_2'. \tag{14}$$

Thus, we conclude that

$$\mathcal{G}_1 \equiv_{\text{func}} \mathcal{G}_2. \tag{15}$$

($\Leftarrow$) If $\mathcal{G}_1 \equiv_{\text{func}} \mathcal{G}_2$, since $\mathcal{G}_1 \preccurlyeq \mathcal{G}_1$ and $\mathcal{G}_2 \preccurlyeq \mathcal{G}_2$, according to *Functional Equivalence Preservation* property, it follows that $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_2 \preccurlyeq \mathcal{G}_1$. □

B Additional Experimental Results

B.1 Functional Subgraph Matching

Considering that the encoder in our method can be replaced with other backbones, we test our approach with different encoders and propose baselines for the functional subgraph detection task, as shown in Table 4.

B.2 Fuzzy Boundary Identification

We further evaluate these methods on fuzzy boundary identification. The results are shown in Table 5.

Table 5: Result of baselines in stage #2.

Method	ITC99		OpenABCD		ForgeEDA	
	IoU	DICE	IoU	DICE	IoU	DICE
Ours+Gamora	82.1	90.2	81.4	89.8	83.6	91.1
Ours+ABGNN	82.7	90.5	84.4	91.5	**88.4**	**93.8**
Ours	**83.0**	**90.7**	**85.2**	**92.0**	83.8	91.2

https://doi.org/10.52202/085713-3336

Table 4: Result of baselines in stage #1.

Dataset	Method	$\mathcal{G}_{sub} \rightarrow \mathcal{G}_{syn}$				$\mathcal{G}_{sub} \rightarrow \mathcal{G}_{pm}$			
		Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
ITC99	Ours+Gamora	90.8	91.1	90.4	90.7	86.4	88.6	83.5	86.0
	Ours+ABGNN	87.9	83.1	95.1	88.7	88.2	82.8	**96.5**	89.1
	Ours	**95.3**	**94.4**	**96.3**	**95.4**	**93.1**	**92.3**	94.2	**93.2**
OpenABCD	Ours+Gamora	90.1	89.6	**90.7**	90.2	**91.0**	89.3	**93.2**	**91.2**
	Ours+ABGNN	81.7	78.5	87.5	82.7	83.3	78.9	91.1	84.5
	Ours	**92.3**	**93.7**	90.6	**92.1**	90.8	**92.4**	88.9	90.6
ForgeEDA	Ours+Gamora	94.2	95.9	92.4	94.1	80.6	93.8	65.5	77.1
	Ours+ABGNN	89.7	88.5	91.2	89.8	87.6	88.3	86.8	87.5
	Ours	**96.0**	**96.8**	**95.2**	**96.0**	**95.3**	**95.9**	**94.7**	**95.3**

B.3 Scalability on Medium-Sized Circuits

we collect a medium-sized graph dataset from ForgeEDA [17], containing circuits with graph sizes ranging from 100 to 10000 nodes. The statistical information of the medium-sized dataset is shown in Table 6.

Table 6: Statistics of the medium-sized dataset.

Graph Type	**Nodes**	**Edges**	**Depth**
G_{sub}	192.1 ± 320.87	207.16 ± 354.01	27.91 ± 30.08
G_{syn}	1396.99 ± 1764.63	1958.84 ± 2519.23	66.48 ± 99.7
G_{pm}	679.93 ± 851.2	1352.96 ± 1703.16	21.83 ± 32.6

We perform functional subgraph detection on this dataset, and the results are shown in Table 7 and 8. Since ABGNN encounters out-of-memory error when encoding graphs with deep logic levels, we do not report its results on this new dataset. While our method still demonstrates state-of-the-art performance, it shows a significant performance drop (from an F1-score of 95.3% to 81.2%, as shown in Table 1). This result highlights the challenge of scaling to larger circuits. We hope future work will explore and address this challenge.

Table 7: Functional Subgraph Detection on $G_{sub} \rightarrow G_{syn}$.

Method	**Accuracy**	**Precision**	**Recall**	**F1-score**
NeuroMatch	$51.2_{\pm 3.3}$	$34.6_{\pm 24.5}$	$66.7_{\pm 47.1}$	$45.5_{\pm 32.2}$
HGCN	$50.0_{\pm 0.0}$	$16.7_{\pm 23.6}$	$33.3_{\pm 47.1}$	$22.2_{\pm 31.4}$
Gamora	$50.0_{\pm 0.0}$	$40.0_{\pm 14.1}$	$66.7_{\pm 47.1}$	$44.6_{\pm 31.3}$
Ours	$\mathbf{81.5_{\pm 0.6}}$	$\mathbf{82.7_{\pm 1.1}}$	$\mathbf{79.8_{\pm 1.7}}$	$\mathbf{81.2_{\pm 0.8}}$

B.4 Comparison with VF3

we evaluate the state-of-the-art subgraph isomorphic heuristic algorithm VF3 [34], which consistently achieves 100% precision and 100% recall on standard subgraph isomorphism tasks. Due to time constraints, we sampled circuits with fewer than 50 nodes from the ForgeEDA dataset and applied the VF3 algorithm. The results are shown in Table 9. According to our Definition 3 of Functional Subgraph, if Q is an isomorphic subgraph of G, then Q is always a functional subgraph of G. This is demonstrated by the 100% precision achieved by VF3. However, due to the function-preserving transformation, the explicit structure of Q often disappears, leading to extremely low recall(0.32%) for VF3. These results highlight the importance of task definition.

Table 8: Functional Subgraph Detection on $G_{sub} \rightarrow G_{pm}$.

Method	Accuracy	Precision	Recall	F1-score
NeuroMatch	$51.0_{\pm 1.2}$	$33.9_{\pm 24.0}$	$66.6_{\pm 47.1}$	$44.9_{\pm 31.8}$
HGCN	$50.0_{\pm 0.0}$	$16.7_{\pm 23.6}$	$33.3_{\pm 47.1}$	$22.2_{\pm 31.4}$
Gamora	$50.0_{\pm 0.0}$	$33.3_{\pm 23.6}$	$66.7_{\pm 47.1}$	$44.4_{\pm 31.4}$
Ours	$\mathbf{78.9_{\pm 1.0}}$	$\mathbf{80.6_{\pm 1.6}}$	$\mathbf{76.3_{\pm 2.7}}$	$\mathbf{78.3_{\pm 1.3}}$

Table 9: Comparison of subgraph isomorphism methods on different tasks.

		$G_{sub} \rightarrow G_{syn}$			$G_{sub} \rightarrow G_{pm}$		
Method	**Runtime (s)**	Precision	Recall	F1-score	Precision	Recall	F1-score
VF3	480.8	**100.0**	0.32	0.65	—	—	—
Ours	**8.0**	88.5	**91.9**	**90.2**	**86.4**	**94.5**	**90.3**

C Datasets and Implementation Details

Dataset Dataset statistics and splits are shown in Table 10. For dataset split, we first split the training circuits and test circuits in the source dataset, then we cut subgraph for the training circuit and test circuits to generate our small circuit dataset. For ITC99 and OpenABCD, the split follow the previous work [24]. For ForgeEDA, we randomly select 10% circuits in the dataset as test circuits. For small circuit, we apply Algorithm 1 to randomly sample subgraph.

Algorithm 1 Random Sample Subgraph

Input: ndoes $\mathcal{V}$, edges $\mathcal{E}$, root r
Output: nodes V, edges E, root r

```
Build adjacency G from E
if rand(0,1) < 0.5 then
    Set r to the predecessor p ∈ G[r] that maximizes predCount(p)
end if
ρ ← rand(0.6, 0.95)
T ← ρ · |V|, V ← {r}, Q ← [r], E ← ∅
while Q ≠ ∅ ∧ |V| < T do
    n ← pop(Q)
    for all v ∈ shuffle(G[n]) do
        if v ∉ V then
            push(Q, v); V ∪ {v}; E ∪ {(n, v)}
        end if
    end for
end while
return V, E, r
```

Table 10: Dataset Statistics. We report average and standard error with $avg. \pm std.$

Source Dataset	Split	#Pair	$\mathcal{G}_{sub}$		$\mathcal{G}_{aig}$		$\mathcal{G}_{syn}$		$\mathcal{G}_{pm}$	
			#Node	Depth	#Node	Depth	#Node	Depth	#Node	Depth
ITC99	train	36592	$248_{\pm 132}$	$15.0_{\pm 2.0}$	$320_{\pm 166}$	$19.1_{\pm 3.0}$	$315_{\pm 164}$	$19.0_{\pm 3.0}$	$179_{\pm 91}$	$6.9_{\pm 1.0}$
	test	5917	$218_{\pm 113}$	$14.0_{\pm 2.0}$	$282_{\pm 141}$	$17.3_{\pm 2.2}$	$278_{\pm 138}$	$17.0_{\pm 2.0}$	$157_{\pm 79}$	$6.3_{\pm 0.9}$
OpenABCD	train	54939	$155_{\pm 113}$	$13.0_{\pm 2.0}$	$203_{\pm 140}$	$16.4_{\pm 3.2}$	$198_{\pm 134}$	$16.0_{\pm 3.0}$	$108_{\pm 75}$	$5.8_{\pm 1.1}$
	test	9726	$100_{\pm 66}$	$13.0_{\pm 2.0}$	$132_{\pm 84}$	$16.0_{\pm 2.2}$	$128_{\pm 82}$	$15.0_{\pm 2.0}$	$69_{\pm 46}$	$5.5_{\pm 0.9}$
ForgeEDA	train	60183	$126_{\pm 102}$	$13.4_{\pm 3.5}$	$161_{\pm 129}$	$16.6_{\pm 4.2}$	$156_{\pm 125}$	$16.2_{\pm 4.5}$	$88_{\pm 69}$	$5.8_{\pm 1.4}$
	test	7753	$127_{\pm 96}$	$13.6_{\pm 3.3}$	$163_{\pm 122}$	$17.0_{\pm 3.8}$	$159_{\pm 120}$	$16.4_{\pm 4.2}$	$89_{\pm 65}$	$5.9_{\pm 1.3}$

https://doi.org/10.52202/085713-3336

Environment All experiments are run on an NVIDIA A100 GPU with 64 GB of memory. Models are trained using the Adam optimizer with a learning rate of 0.001, a batch size of 1024. We train our model in stage#1 for 100 epochs and finetune it in stage#2 for 10 epochs. Training our model on one dataset takes approximately 10 hours. Model architectures follow the configurations specified in the original works except that we set the hidden dimension to 128 for all models.

Evaluation Metrics For Stage #1, we measure classification performance by accuracy and report precision, recall and f1-score according to the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \ \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \ \text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

For Stage #2, which is similar to a segmentation task, we use Intersection over Union (IoU) and the Dice coefficient. Let P be the set of predicted positive nodes and G the set of ground-truth positive nodes:

$$\text{IoU} = \frac{|P \cap G|}{|P \cup G|}, \ \text{Dice} = \frac{2\,|P \cap G|}{|P| + |G|} \tag{16}$$

D Background

And-Inverter Graph(AIG) In our works, AIG is a directed acyclic graph (DAG) composed of three basic elements: Primary Input(PI), AND gate and NOT gate. For example, a simple logic expression $\neg A \wedge B$ can be build as a DAG with 2 PIs(A and B), one NOT gate and one AND gate. The edges are $[(A, NOT), (NOT, AND), (B, AND)]$. Since the out-degree of AND is zero, it represents the final output.

Technology Mapping Technology Mapping is a function-preserving transformation that converts an AIG into a post-mapping (PM) netlist. While the AIG consists of simple logic elements, such as AND and NOT gates, the basic components in a PM netlist can be more complex, such as adders or multipliers. As a result, node types can differ significantly between the two forms, and this is why we consider AIG and PM netlists as distinct modalities in this paper.

Logic Synthesis Logic synthesis aims to simplify the structure of an AIG while preserving its functionality. It transforms one AIG into another with a simpler structure. For example, the expression $eq_1 : (A \wedge B) \wedge (A \wedge C)$ can be simplified to $eq_2 : A \wedge B \wedge C$. Although eq_1 and eq_2 are functional equivalent, eq_2 uses only 2 AND gates compared to 3 AND gates in eq_1.

InfoNCE Loss InfoNCE (Information Noise-Contrastive Estimation) is a contrastive loss used in self-supervised learning. Its goal is to identify a single "positive" sample from a set of "negative" samples for a given "anchor" sample. It pulls the anchor and positive representations together while pushing the anchor and negatives apart:

$$\mathcal{L}_{InfoNCE} = -\log \frac{\exp(\text{sim}(q, k_+)/\tau)}{\sum_{i=0}^{N} \exp(\text{sim}(q, k_i)/\tau)} \tag{17}$$

where q is the anchor, k_+ is the positive, k_i are the negatives, sim is a similarity function (like dot product), and τ is a temperature hyperparameter.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We list our contribution in introduction, as shown in Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of our method in Section 5, including the Scalability to Large-scale Circuits, Multiple and Overlapping Fuzzy Boundaries and Assumptions.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

https://doi.org/10.52202/085713-3336

Justification: We proposed 4 properties of functional subgraph in our paper, and the proof can be found in Appendix A. We also clairfy our assumption in Section 2.3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: For dataset processing, we provide the details in Section 4.1 and Appendix C. For our models, we first detailed our pipeline in Section 3. Futhermore, our used backbones are all opensource and the implementations details can also be found in Section 4.1 and Appendix C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the code and data in our supplemental material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the experimental details in Section 4.1 and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report 1-sigma error bars, i.e. average result with standard error in Table 1 and 2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

https://doi.org/10.52202/085713-3336

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The detail about compute resources can be found in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We make sure research conducted in the paper conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts in Section 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original paper or website of assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

 https://doi.org/10.52202/085713-3336

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We detail our dataset processing flow and model in our paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

https://doi.org/10.52202/085713-3336