MILS: Modality Interaction Driven Learning for Logic Synthesis

Mingyu Zhao
Beijing University of Posts and Telecommunications
Beijing, China
zhaomingyu@bupt.edu.cn

Jiawei Liu
Beijing University of Posts and Telecommunications
Beijing, China
liu_jiawei@bupt.edu.cn

Jianwang Zhai*
Beijing University of Posts and Telecommunications
Beijing, China
zhaijw@bupt.edu.cn

Chuan Shi*
Beijing University of Posts and Telecommunications
Beijing, China
shichuan@bupt.edu.cn

Abstract

Logic synthesis is a key stage in electronic design automation (EDA), where machine learning (ML) techniques are increasingly applied to accelerate the process, particularly in predicting the quality of results (QoR). In the logic synthesis process, each step's synthesis result depends on the previous circuit state and the currently applied synthesis operator. However, existing methods typically encode circuit structures and synthesis sequences separately and concatenate them for prediction, overlooking the interaction between these two modalities. This limitation hinders the model's understanding of the synthesis process and degrades prediction performance. Inspired by this, we propose MILS - a Modality Interaction Driven Learning approach for Logic Synthesis. Firstly, it employs a modality interaction module to capture better and deeper relationships between circuit structures and synthesis sequences, enhancing the modeling capability of the synthesis process. Moreover, MILS designs a permutation-aware self-supervised task to make better use of limited data and further enhance the interaction process by cross-modal contrastive learning. Experimental results on two QoR tasks (i.e., area and delay prediction) show MILS outperforms the state-of-the-art (SOTA) methods, with an improvement of 8.41% and 9.28% in learning capability on two tasks.

CCS Concepts

• **Hardware** → **Logic synthesis**.

Keywords

Logic Synthesis, QoR Prediction, Multimodal Learning

ACM Reference Format:
Mingyu Zhao, Jiawei Liu, Jianwang Zhai, and Chuan Shi. 2025. MILS: Modality Interaction Driven Learning for Logic Synthesis. In *Great Lakes Symposium on VLSI 2025 (GLSVLSI '25), June 30–July 02, 2025, New Orleans, LA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3716368.3735203

*Corresponding authors.

GLSVLSI '25, New Orleans, LA, USA

ACM ISBN 979-8-4007-1496-2/25/06
https://doi.org/10.1145/3716368.3735203

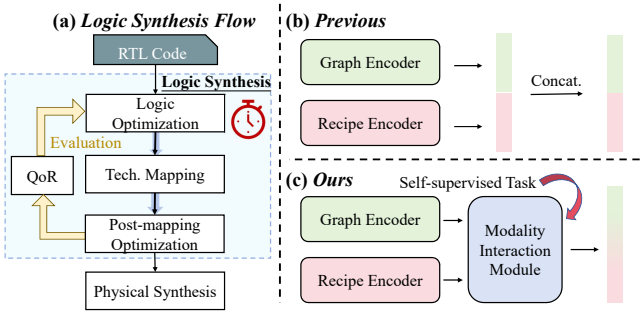

Figure 1: Illustration of the logic synthesis flow and model comparisons on the QoR prediction task.

1 Introduction

EDA is the foundation stone for integrated circuit (IC) design, spanning various stages such as design, synthesis, and verification. Logic synthesis is an essential step in the EDA process, consisting of three main steps: logic optimization, technology mapping, and post-mapping optimizations. This step applies a sequence of heuristic operators, known as a recipe, to synthesize circuits into gate-level netlists. The resulting circuit is evaluated based on key metrics, named quality of results (QoR), such as area and delay, which must meet predefined constraints. If any metric fails to satisfy the constraints or determined optimization objectives, iterative optimization is required until a fully compliant circuit is generated, as illustrated in Figure 1(a).

However, traditional heuristic-based methods struggle with efficient optimization[4]. In this situation, machine learning offers faster and potentially more accurate QoR predictions[5], which significantly enhances the efficiency of the synthesis process. Moreover, the search space of the synthesis sequence grows exponentially as the number of operators and parameter choices increases. By leveraging QoR prediction models, designers can efficiently estimate the impact of different synthesis choices, allowing for intelligent exploration and optimization of operator sequences to achieve superior chip design [14, 17].

The quality of the circuit after each synthesis step depends on both the previous circuit state and the current operator. In recent years, many machine learning-based methods have been applied to predict circuit quality[2, 3, 13, 16, 19]. However, these works **overlook the interaction between the two modalities**. Specifically, these methods convert circuits into and-inverter graphs (AIGs), encode synthesis sequences, and use graph neural networks (GNNs) to model the circuit while employing sequence models to represent the

recipe. The final prediction is then made based on the concatenated representations of both modalities, as illustrated in Figure 1(b). This paradigm of separate modeling and direct concatenating ignores the interaction between two modalities and treats a sequence of ordered operators as a single entity, which hinders the model's ability to understand the logic synthesis process. Therefore, capturing the interaction between these modalities is crucial for understanding the logic synthesis process and improving model accuracy.

To address these problems, we propose MILS - a Modality Interaction Driven Learning approach for Logic Synthesis, as shown in Figure 1(c). Firstly, inspired by the logic synthesis process, it employs a modality interaction module to capture better and deeper relationships between circuit structures and synthesis sequences, enhancing the modeling capability of the synthesis process. Moreover, based on the permutation-variant property of synthesis recipes, MILS designs a permutation-aware self-supervised task to make better use of limited data and further enhance the interaction process by cross-modal contrastive learning. This task can effectively supervise intermediate states between interactions without relying on labeled data. The main contributions are as follows:

- **Study modality interaction in logic synthesis**: We first investigate the role of modality interaction in logic synthesis process, offering new insights for model design in this field.
- **Propose a modality interaction module**: We introduce a modality interaction module that models the interaction between different modalities at a fine-grained level, enabling the model to better capture cross-modal relationships and improve predictive performance.
- **Introduce a permutation-aware self-supervised task**: Considering the characteristics of recipes, we design a permutation-aware self-supervised task to better utilize limited data while supervising the hidden states in the interaction.
- **Extensive experiments on QoR prediction**: We conduct comprehensive experiments on QoR prediction, demonstrating that MILS effectively models modality interactions and significantly improves accuracy and generalization.

2 Preliminaries

In this section, we systematically review related SOTA research and formally give the problem definition. In Section 2.1, we revisit prior ML-based approaches for circuit design quality prediction and analyze their limitations. In Section 2.2, we introduce research on modality fusion in multimodal learning, exploring how different modalities interact. In Section 2.3, we formally characterize the modality interactions in the logic synthesis process and discuss the permutation variability of synthesis recipes.

2.1 Logic Synthesis and QoR Prediction

Logic synthesis is a crucial step in the EDA flow, transforming a hardware description language program into a functionally equivalent gate-level netlist. AIG is a common representation that uses two-input AND nodes and inverter edges for efficient optimization. The synthesis process optimizes the circuit through a sequential decision process applying sub-graph optimization heuristics in a non-trivial combination. The quality of designs at each step relies on the designs from the previous step and the heuristics used in the current step. The goal of logic synthesis is to result in a circuit satisfying QoR in terms of area, delay, and power consumption after technology mapping. However, traditional methods are time-consuming and require intensive manual efforts.

With the advancement of ML, numerous ML-based methods have been proposed to predict the quality of circuits after logic synthesis. These methods typically leverage synthesis tools to transform RTL code into an unoptimized intermediate representation, which is then used for downstream tasks such as QoR prediction. To effectively utilize this intermediate representation, various circuit representation learning techniques [9, 11, 12] are employed to capture the essential properties of the circuit. For instance, in the context of these QoR prediction approaches, LOSTIN[16] treats the circuit as spatial information, using a GNN for modeling, while the synthesis sequence is treated as temporal information, processed with an LSTM. The two modalities are then concatenated for final prediction. OpenABC-D[2] employs a GNN for AIG modeling and 1D convolutional networks for sequence processing while also providing an open-source dataset. LSTP[19] designs a circuit sampling algorithm and employs a transformer-based sequence model for circuit performance prediction, with a focus on delay estimation. However, this approach of modeling the two modalities separately fails to capture their interactions, limiting the model's comprehension of the synthesis process and reducing prediction accuracy.

2.2 Modality Fusion

Multimodal learning has achieved remarkable success in various fields, such as vision-language[7, 8, 15] and graph-language learning [10, 18]. For example, ALBEF[8] proposes a two-stage framework that first aligns unimodal representations of vision and language and then uses the cross-attention mechanism to capture fine-grained information between different modalities. Building upon this, BLIP[7] introduced a multimodal mixture of encoder-decoder architecture, which unifies vision-language understanding and generation tasks. In another domain, OFA[10] handles graph and text modalities by representing graph nodes and edges with natural language and then using language models to create unified feature vectors.

In the multimodal learning process, the fusion strategy plays a crucial role in capturing cross-modal relationships and extracting meaningful information[6]. Each approach has distinct advantages: for instance, late fusion effectively captures comprehensive information from different modalities independently, while intermediate fusion preserves modality-specific features while enabling efficient cross-modal interactions. Most existing logic synthesis prediction methods rely on late fusion, which overlooks cross-modal interactions. Unlike conventional vision-language tasks, the interaction between graph and sequence modalities in logic synthesis follows a clear relationship, making the direct application of existing fusion techniques unsuitable.

2.3 Problem Formulation

Given an AIG $G = \{V, E\} \in \mathcal{G}$, where $v_i \in V$ represents the circuit gate node, and $e_{ij} \in E$ represents the directed edge from gate node v_i to v_j. For a K-length sequence $r = (h_1, h_2, \ldots, h_K) \in \mathcal{R}$, logic synthesis can be described as the sequential process:

$$s_0 \xrightarrow{h_1} s_1 \xrightarrow{h_2} s_2 \xrightarrow{h_3} \cdots \xrightarrow{h_K} s_K, \tag{1}$$

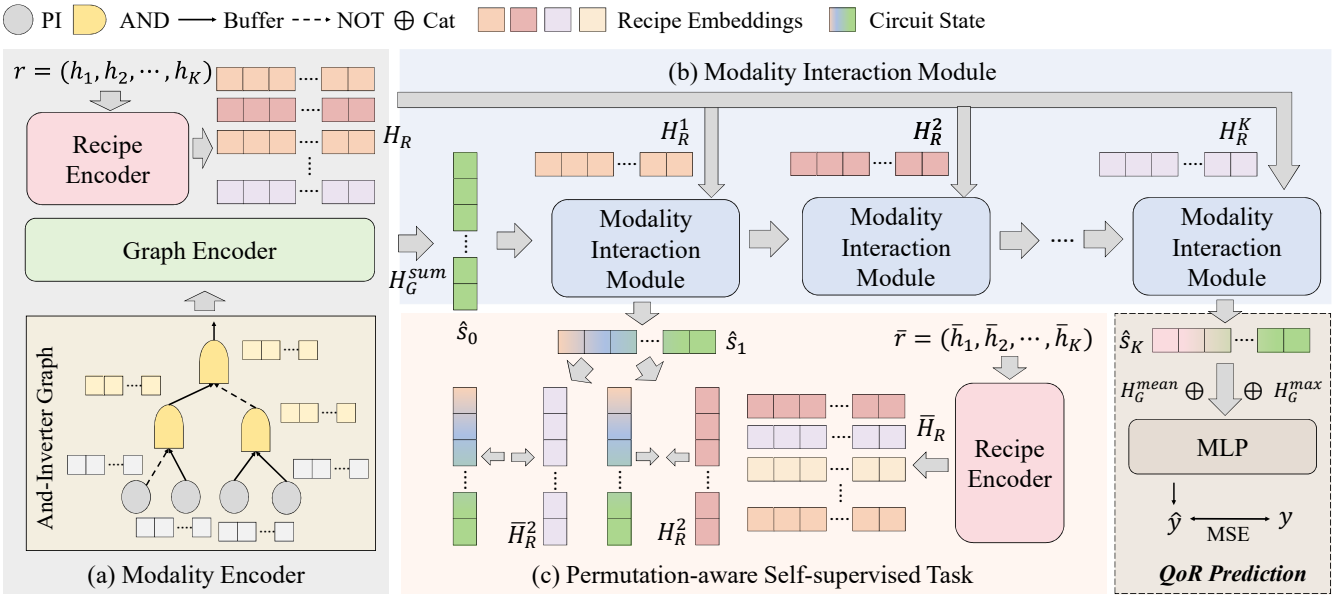

Figure 2: Overview of MILS: (a) Modality Encoder. Encode circuit structures and synthesis recipes. (b) Modality Interaction Module. Model the sequential interaction between circuit states and synthesis steps. (c) Permutation-aware Self-supervised Task. Leverage the permutation-aware property of recipes to supervise intermediate states.

where each recipe $h_i \in r$ is chosen from a set $\mathcal{T} = \{t_1, t_2, \ldots, t_C\}$ of C operators, and $\{s_i\}_{i=0}^{K}$ represents the evolving circuit states after each operator. This process demonstrates that the result of each step relies on the previous circuit state and the currently applied synthesis operator.

Additionally, the synthesis sequence is ordered and permutation-variant, which means that different operator orders produce different synthesis results. Formally, for a permutation $\epsilon(\cdot)$, the permuted recipe is $\epsilon(r) = (h_{\epsilon(1)}, h_{\epsilon(2)}, \ldots, h_{\epsilon(K)})$, and the synthesis results mostly satisfy:

$$f(G, r) \neq f(G, \epsilon(r)), \tag{2}$$

where f denotes the mapping function. Also, there's often an implicit dependency, which highlights that the effect of each operator is critically conditioned by the sequence of preceding operators and the specific circuit state they establish. This property drives our approach to improving existing models by better capturing the modality interactions in logic synthesis.

In summary, logic synthesis prediction aims to learn the function:

$$f : \mathcal{G} \times \mathcal{R} \rightarrow \mathbb{R}, \tag{3}$$

where $\mathcal{G}$ is the set of AIGs and $\mathcal{R}$ is the set of synthesis recipes.

3 Methodology

Inspired by the interaction process and permutation-aware characteristics, we propose a novel logic synthesis prediction framework called MILS. As shown in Figure 2, MILS consists of three components: modality encoder, modality interaction module, and permutation-aware self-supervised task. Specifically, MILS first encodes the circuit and synthesis sequence separately to obtain two distinct representations in Figure 2(a). These representations are then fed into the modality interaction module in Figure 2(b), where the information from both modalities interacts extensively. Finally, MILS uses the permutation-aware self-supervised task without the need for additional labels in Figure 2(c). The final representations obtained from MILS can then be used to predict the synthesis quality metrics of the circuit, such as area and delay.

3.1 Modality Encoder

3.1.1 Graph Encoder. Graph neural networks are powerful tools for processing graph-structured data, effectively capturing complex relationships and dependencies between nodes. By iteratively aggregating information from neighboring nodes, GNNs learn both local connectivity patterns and global structural properties. Generally, a GNN layer typically consists of two stages: aggregation and combination. For a node v with hidden embedding $x_v^{(l-1)}$ at layer l, this process is formalized as:

$$\begin{aligned} m_v^{(l)} &= AGGREGATE^{(l)}(\{x_u^{(l-1)} : u \in \mathcal{N}(v)\}), \\ x_v^{(l)} &= COMBINE^{(l)}(x_v^{(l-1)}, m_v^{(l)}), \end{aligned} \tag{4}$$

where $\mathcal{N}(v)$ denotes the neighbors of v. This process ensures permutation invariance in aggregation and effective fusion of local and global circuit properties. Since circuits are naturally graph structures, GNNs can learn both local connectivity patterns and global structural properties essential for QoR prediction in logic synthesis.

Logic-level designs are converted into AIGs, where nodes represent logic gates and edges denote signal propagation[2]. For a circuit graph $G = \{V, E\}$, the initial node features $X \in \mathbb{R}^{|V| \times D_0}$ typically encode gate types, node degrees, and other circuit characteristics. After processing through an L-layer GNN, the updated node features $X^L \in \mathbb{R}^{|V| \times D}$ are obtained as:

$$X^L = GNN(G, X). \tag{5}$$

To obtain a global circuit representation, graph pooling techniques aggregate node embeddings into a fixed-dimensional vector,

balancing local details and global circuit behavior. Different pooling strategies emphasize different aspects of the circuit:

$$\begin{aligned} H_G^{max} &= MAX\_Pool(X^L) \rightarrow \text{Capture dominant features,} \\ H_G^{mean} &= MEAN\_Pool(X^L) \rightarrow \text{Preserve average behavior,} \\ H_G^{sum} &= SUM\_Pool(X^L) \rightarrow \text{Reflect total activity,} \end{aligned} \tag{6}$$

where $H_G^{max}, H_G^{mean}, H_G^{sum} \in \mathbb{R}^{1\times D}$. This approach effectively captures overall circuit characteristics.

3.1.2 Recipe Encoder. Each logic synthesis sequence consists of a series of operators arranged in a specific order. For a K-length recipe $r = (h_1, h_2, \ldots, h_K) \in \mathcal{R}$, we employ learnable embeddings to capture the characteristics of each operator within the sequence. These embeddings allow the model to learn a dense representation of the operators and their interactions within the sequence:

$$H_R = Embedding(r), \tag{7}$$

where $H_R \in \mathbb{R}^{K\times F}$ represents the embedding matrix, with each row corresponding to the learnable feature representation of an individual operator in the sequence. This approach enables the model to effectively capture both the inherent properties of the operators and their sequential relationships.

3.2 Modality Interaction Module

In the logic synthesis process, each step's synthesis result depends on the previous circuit state and the currently applied synthesis operator, forming a sequential dependency. To effectively model this progress while maintaining computational efficiency, we employ the Gated Recurrent Unit (GRU) [1]. Specifically, MILS learns interactions through four steps as follows.

3.2.1 Circuit State Initialization. To establish an initial circuit representation, MILS employs sum-pooling over the graph representation:

$$\hat{s}_0 \triangleq H_G^{sum} \in \mathbb{R}^{1\times D}. \tag{8}$$

This initialization captures the overall structural information of the circuit before any synthesis operation is applied.

3.2.2 Selective Filtering of Previous Circuit Information. Since different synthesis operators influence the circuit differently, it is essential to selectively filter relevant features from the previous state. Formally, at the k-th synthesis step, this is achieved through a gating mechanism:

$$r_k = \sigma(H_R^k W_{hr} + \hat{s}_{k-1} W_{sr} + b_r), \tag{9}$$

where $W_{hr} \in \mathbb{R}^{F\times D}$ and $W_{sr} \in \mathbb{R}^{D\times D}$ are transformation matrices, $b_r \in \mathbb{R}^{1\times D}$ is a bias term, and $\sigma(\cdot)$ is an activation function (e.g., sigmoid). A lower value of r_k indicates that more irrelevant circuit features are filtered out, preserving only useful information for the current synthesis operator and making modality interaction more expressive.

3.2.3 Candidate Circuit State Generation. Once relevant features are selected, MILS updates the candidate circuit state $\tilde{s}_k \in \mathbb{R}^{1\times D}$:

$$\tilde{s}_k = \tanh(H_R^k W_{hs} + (r_k \odot \hat{s}_{k-1}) W_{ss} + b_s), \tag{10}$$

where $W_{hs} \in \mathbb{R}^{F\times D}$ and $W_{ss} \in \mathbb{R}^{D\times D}$ are transformation matrices, $b_s \in \mathbb{R}^{1\times D}$ is a bias term, and $\odot$ represents the Hadamard product. This step ensures that the new candidate state effectively incorporates both the filtered circuit features $r_k \odot \hat{s}_{k-1}$ and the synthesis operator H_R^k, enhancing the interaction between modality representations.

3.2.4 Balancing Historical and Current Circuit Information. However, considering only the current operation may cause the model to forget important historical optimization states. To address this, MILS balances the previous state and the candidate state using a weighted sum:

$$\begin{aligned} z_k &= \sigma(H_R^k W_{hz} + \hat{s}_{k-1} W_{sz} + b_z), \\ \hat{s}_k &= z_k \odot \hat{s}_{k-1} + (1 - z_k) \odot \tilde{s}_k. \end{aligned} \tag{11}$$

Here, $W_{hz} \in \mathbb{R}^{F\times D}$ and $W_{sz} \in \mathbb{R}^{D\times D}$ are transformation matrices, and $b_z \in \mathbb{R}^{1\times D}$ is a bias term. The dynamic weight mechanism adjusts how much the candidate state $\tilde{s}_k$ affects the final circuit state $\hat{s}_k$, ensuring essential past information is preserved while allowing for important updates.

After processing all K synthesis steps, we apply a multi-layer perceptron (MLP) to map the final hidden state to predicted QoR:

$$\hat{y} = MLP(Concat(\hat{s}_K, H_G^{max}, H_G^{mean})). \tag{12}$$

This design ensures that both sequential synthesis effects and structural circuit characteristics contribute to accurate QoR estimation.

The modality interaction module (MIM) offers several advantages. First, it explicitly captures the interaction between circuit structure and synthesis sequences, ensuring that the evolution of circuit states is effectively modeled. Second, by maintaining a dynamically updated representation, MIM integrates sequential synthesis steps with structural information, allowing the model to learn how different operations collectively impact the circuit. Lastly, its design aligns well with the progressive nature of logic synthesis, enabling a more informed and adaptive prediction of circuit performance.

3.3 Permutation-aware Self-supervised Task

Although the modality interaction module enhances the model's ability to capture interactions between the circuit and synthesis recipe, the hidden circuit state s_i is only conditioned on the previous circuit state s_{i-1} and the current synthesis operator h_i. Without explicit intermediate supervision, this representation may lack sufficient guidance towards configurations that are not only optimal so far, but also amenable to subsequent effective synthesis operations. This could lead to a suboptimal understanding and modeling of the multi-stage synthesis process.

A key characteristic of effective logic synthesis recipes is that operators are not arbitrarily ordered; rather, there's often an implicit dependency or suitability between a circuit state and the subsequent operations. An operator h_{i+1} is typically chosen or is effective because the circuit, after transformation by $h_1, \ldots, h_i$ (resulting in state s_i), is in a condition that makes h_{i+1} a pertinent next step. Inspired by this, we introduce a permutation-aware auxiliary loss that leverages the next synthesis operator to supervise the learning of the previous circuit state.

Specifically, for each $r \in \mathcal{R}$, we randomly generate negative recipe $\bar{r} = (\bar{h}_1, \bar{h}_2, \ldots, \bar{h}_K)$. We then obtain its corresponding embedding $\bar{H}_R$. Given a sequence of hidden circuit states $\{\hat{s}_i\}_{i=1}^{K-1}$ and original recipe embedding H_R, auxiliary loss is formulated as:

$$\mathcal{L}_{aux} = -(\sum_{i=1}^{K-1} \log \gamma(\hat{s}_i, H_R^{i+1}) + \log(1 - \gamma(\hat{s}_i, \bar{H}_R^{i+1}))), \tag{13}$$

where $\gamma(x_1, x_2) = \frac{1}{1+\exp(-[x_1,x_2])}$ is sigmoid activation function.

This contrastive formulation provides two key benefits. **Enhanced State Representation for Sequential Decision Making.** It encourages alignment between hidden states and the correct synthesis progression while discouraging alignment with incorrect operator sequences, improving representation learning of states that are good precursors for subsequent steps. **Intermediate Supervision without Explicit Labels.** It leverages the sequential property of synthesis recipes to provide supervision without requiring intermediate-step labels, effectively alleviating data limitation issues.

In summary, our primary regression loss for QoR prediction is:

$$\mathcal{L}_{target} = RegressionLoss(y, \hat{y}), \tag{14}$$

where y represents the ground-truth QoR metric, and $\hat{y}$ is the model's predicted value. Then, the total loss function is as follows:

$$\mathcal{L} = \mathcal{L}_{target} + \alpha \mathcal{L}_{aux}, \tag{15}$$

where α is the hyper-parameter controlling the contribution of the auxiliary supervision.

By incorporating this permutation-aware auxiliary loss, our framework cultivates more informative intermediate circuit state representations. These representations are better attuned to the sequential nature of the synthesis process, leading to a more nuanced understanding of how operator sequences transform circuits and ultimately improving the accuracy of final QoR prediction.

4 Evaluation

We conduct a comprehensive experimental evaluation to demonstrate the effectiveness of MILS in QoR prediction. Our study is guided by the following key research questions:

(1) Does MILS effectively capture the interaction between circuit structure and synthesis recipes, leading to improved QoR prediction performance?
(2) How does MILS perform on circuits with varying functionalities and scales?
(3) How do different hyperparameter settings affect MILS's performance?
(4) Does each component of MILS contribute positively to the final prediction results?

We first introduce the experimental setup in Section 4.1, and then sequentially address the aforementioned questions. To answer the first question, we test MILS on circuit area and delay prediction in Section 4.2, demonstrating its ability to leverage multimodal information. To respond to the second question, we present a case study in Section 4.3 to examine MILS's performance on circuits with diverse functionalities and scales. We address the third question with hyper-parameter experiments in Section 4.4. Finally, in Section 4.5, we perform an ablation study on MILS.

4.1 Experimental Setup

We implemented extensive experiments to assess the performance of MILS. These We conduct extensive experiments to evaluate the effectiveness of MILS. All experiments are performed on an NVIDIA GeForce RTX 3090 GPU. The software environment includes CUDA Driver 12.7, PyTorch v1.12.0, and PyG v1.7.0.

Table 1: The Statistics of Datasets.

Circuit Name	#Node	#Level	#Area	#Delay
spi	4219	35	[1925.57-2409.43]	[1470.34-3201.25]
i2c	1169	15	[618.72-712.88]	[433.81-961.11]
ss_pcm	462	10	[259.62-310.95]	[380.97-880.00]
usb_phy	487	10	[308.56-340.75]	[208.32-358.25]
sasc	613	9	[411.50-525.08]	[335.05-732.60]
wb_dma	4587	29	[2351.17-2792.73]	[1379.07-3803.23]
simple_spi	930	12	[536.26-676.44]	[333.82-665.83]
pci	19547	29	[11656.39-14313.99]	[4814.33-14264.05]
wb_conmax	47840	24	[24817.80-26912.82]	[2114.13-4581.00]
ethernet	67164	34	[40771.95-49102.80]	[6639.66-22833.54]
dynamic_node	18094	33	[10902.81-13353.73]	[1390.67-7659.31]
ac97_ctrl	11464	11	[7363.41-9028.04]	[632.47-2155.95]
mem_ctrl	16307	36	[5854.13-8081.88]	[2261.74-5168.72]
bp_be	82514	86	[47169.78-56543.09]	[11195.65-85487.38]
vga_lcd	105334	23	[63257.20-82467.98]	[9684.69-66028.22]
des3_area	4971	30	[3544.98-3673.99]	[2972.50-3215.63]
aes	28925	27	[16997.40-20432.26]	[868.44-3697.58]
sha256	15816	76	[9400.97-10321.33]	[10138.81-154699.81]
aes_xcrypt	45840	43	[30371.61-33248.94]	[18903.64-199815.28]
aes_secworks	40778	42	[21339.85-25411.25]	[7874.45-25751.47]
fir	4558	47	[3475.56-3680.38]	[1019.26-1220.51]
iir	6978	73	[4713.25-5039.90]	[1678.86-1955.74]
jpeg	114771	40	[86808.57-92175.91]	[13358.58-19807.41]
tv80	11328	54	[5954.14-6568.87]	[2329.29-3313.84]
tinyRocket	52315	80	[27132.27-30245.53]	[9535.28-50530.97]
fpu	29623	819	[19132.58-20029.80]	[29484.68-33453.33]
picosoc	82945	43	[46504.25-56754.56]	[6778.78-65097.03]
Total	**[462-114771]**	**[9-819]**	**[259.62-92175.91]**	**[208.32-199815.28]**

4.1.1 Evaluation Tasks and Metrics. To comprehensively evaluate the performance of MILS, we focus on two key QoR prediction tasks: area and delay, which are critical metrics in logic synthesis[2, 16]. For evaluation, we use Mean Absolute Error (MAE) and Mean Squared Error (MSE), which are commonly used metrics in regression tasks. MAE measures the average absolute difference and provides an intuitive measure of overall prediction accuracy. MSE, by penalizing larger errors more heavily, captures variance and aligns with the optimization objective.

Specifically, for all labels $\{y_i\}_{i=1}^N$ and prediction results $\{\hat{y}_i\}_{i=1}^N$, MAE and MSE are defined as:

$$\begin{aligned} MAE &= \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|, \\ MSE &= \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2. \end{aligned} \tag{16}$$

4.1.2 Datasets and Baselines. We use the OpenABC-D dataset[2] for training and evaluation. It aggregates IP designs from MIT LL Labs CSP, OpenCores, and IWLS, optimized with Yosys and ABC. The statistics of the dataset are presented in Table 1. As indicated in Table 1, the circuit dataset has a wide range of circuit sizes, varying from hundreds to thousands of nodes with different logic levels. The dataset shows significant variability in both area and delay, which provides rich scenarios for evaluating a model's generalization under various logic recipes. Compared to ISCAS and EPFL, OpenABC-D features more complex and diverse circuits, providing a more rigorous benchmark for model generalization. Our experiments involve

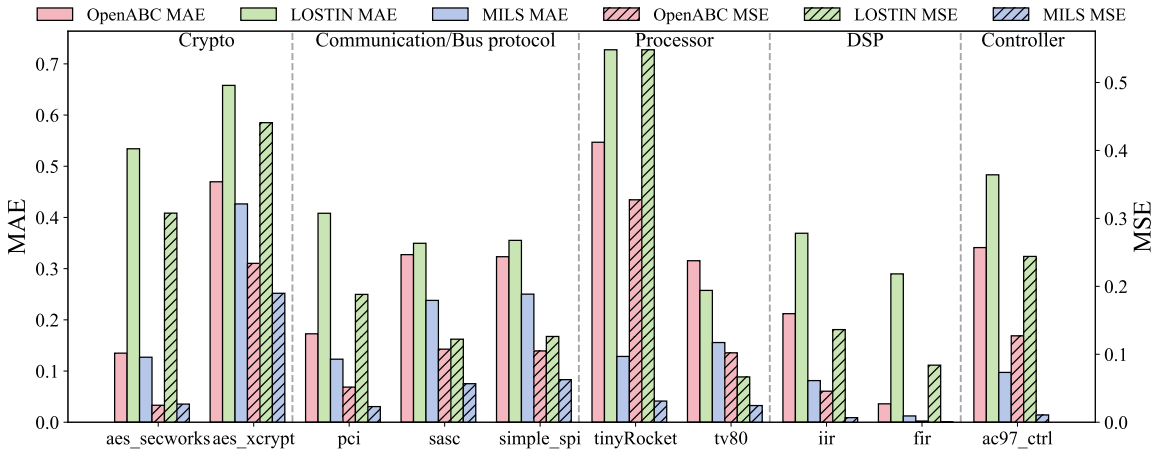

Figure 3: Case Study: compare MAE and MSE of area prediction across baselines.

Table 2: Accuracy results on area and delay tasks.

Method	Area		Delay	
	MAE	MSE	MAE	MSE
OpenABC [2]	0.3783	0.3467	0.9597	3.2339
LOSTIN [16]	0.4467	0.2916	0.9756	3.8302
MILS (Ours)	**0.3465**	**0.2856**	**0.8706**	**3.1576**

27 circuits and 1,500 synthesis sequences, using a 4/1/95 data split: 5% of the synthesis sequences serve as the training set, further divided into training and validation sets at an 8:2 ratio. This setup enables a robust evaluation of the model's performance under limited training data and its generalization ability.

We select OpenABC [2] and LOSTIN [16] as baselines for area and delay prediction due to their strong performance, open-source availability, and widespread use in comparative studies. For a fair comparison, the hyperparameter settings are kept consistent with their original papers. Our graph encoder uses ten layers of GIN with a hidden embedding size of 32. The recipe encoder's embedding is set to 20. Moreover, the hyperparameter α is set to 0.5.

4.2 Accuracy Evaluations

We focus on predicting a circuit's area and delay under unseen synthesis recipes. Notably, our training set is significantly smaller than the test set, which strongly indicates the model's generalization ability. As shown in Table 2, MILS consistently outperforms SOTA methods across all evaluation metrics, achieving the lowest MAE and MSE for both tasks. These improvements demonstrate the model's capability to effectively capture the interaction between circuit structure and synthesis recipes, leading to more accurate and reliable QoR predictions.

Specifically, MILS achieves an MAE of 0.3465 and MSE of 0.2856 for the area task, improving upon the SOTA model by 8.41% and 2.06%, respectively. Similarly, for the delay task, MILS attains an MAE of 0.8706 and MSE of 3.1576, reducing errors by 9.28% and 2.36%. These results suggest that MILS effectively learns cross-modal representations, making it a more robust and adaptable model for logic synthesis QoR prediction.

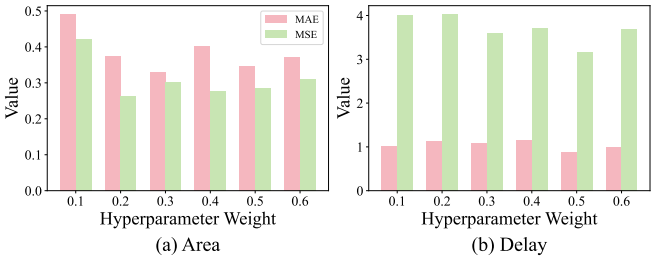

Figure 4: Performance with different hyperparameter weight.

4.3 Case Study

To further analyze the performance of models, we conduct a case study on 10 representative circuits, as shown in Figure 3. The selection follows three key principles: 1) Functional diversity: cover five core categories (crypto, communication/bus protocol, etc.); 2) Scale coverage: ensuring a wide range of node counts (613–52,315) and depths (9–80); 3) Industrial relevance: prioritize widely used designs (e.g., tinyRocket processor, ac97_ctrl controller). Due to space limitations, we focus on the area prediction task, comparing MAE and MSE across OpenABC, LOSTIN, and MILS.

Experimental results show that MILS consistently outperforms OpenABC and LOSTIN across circuits of different functions and scales. Functionally, MILS achieves the lowest MAE and MSE across all circuit categories, particularly excelling in processor and controller circuits. For instance, in tinyRocket, MILS significantly outperforms both baselines, highlighting its advantage in handling complex designs. In terms of scale, MILS maintains stable performance across both small circuits (sasc, 613 nodes) and large circuits (tinyRocket, 52K nodes), with MAE variations kept below 0.3. This highlights MILS's ability to handle both high-complexity and low-resource designs effectively. Overall, MILS effectively learns the interaction between circuit structure and synthesis recipes, achieving strong generalization across different functionalities and scales.

4.4 Impact of Hyper-parameter

Figure 4 illustrates the impact of different hyperparameter weights on MILS's performance for area and delay prediction tasks, where the x-axis represents the weight values and the y-axis shows the corresponding MAE (red) and MSE (green).

Table 3: Ablation results on area and delay tasks.

MIM	SSL	Area		Delay	
		MAE	MSE	MAE	MSE
✗	✗	0.7218	0.9189	1.0464	3.5914
✔	✗	**0.3293**	0.3485	1.1979	3.6275
✔	✔	0.3465	**0.2856**	**0.8706**	**3.1576**

On the one hand, for area, both MAE and MSE show a decreasing trend from 0.1 to 0.3, indicating an improvement in prediction accuracy. However, beyond 0.3, the values remain stable or even increase slightly, suggesting that excessive emphasis on the auxiliary loss might lead to over-regularization or the loss of useful information. On the other hand, for delay, MAE remains stable across all weight values, while MSE decreases from 0.3 to 0.6, indicating that the auxiliary loss contributes more to variance reduction rather than absolute error minimization. Compared to the area task, the improvements for delay are less significant, suggesting that further refinements in model design may be necessary to enhance performance. Overall, these observations highlight the need for task-specific hyperparameter tuning, as different tasks benefit from different weighting strategies, emphasizing the importance of selecting appropriate parameters for each prediction objective.

4.5 Ablation Study

To verify the effectiveness of each module in MILS, we present the results of the ablation study in Table 3, evaluating the impact of the modality interaction module (MIM) and self-supervised learning (SSL). To understand the contribution of each module, we conduct experiments by selectively removing them. Removing MIM means that the synthesis sequence is directly fed into the GRU, and its final representation is concatenated with the graph representation for prediction. Removing SSL indicates that no auxiliary self-supervised loss is applied during training, meaning the model relies solely on the primary regression loss for optimization.

From the results, several key observations can be made. First, removing both MIM and SSL results in the worst performance across all metrics, indicating that both components play essential roles in enhancing the model's predictive capabilities. Second, when MIM is included but SSL is removed, the model achieves a significant improvement in area prediction, reducing MAE to 0.3293 and MSE to 0.3485. However, for delay prediction, performance deteriorates, suggesting that MIM alone is insufficient for optimizing all tasks, particularly delay estimation. Finally, incorporating both MIM and SSL leads to the best overall performance, which confirms that the combination of both components provides the most robust and balanced performance, underscoring the necessity of integrating structural information and self-supervised learning into MILS.

5 Conclusion

In this work, we analyze the logic synthesis process and identify that existing models overlook the interaction between circuits and synthesis sequences. Inspired by this, we propose MILS, a modality interaction driven learning framework that introduces a modality interaction module and a permutation-aware self-supervised task tailored to synthesis sequences. MILS employs a GRU to iteratively update circuit states, capturing the dynamic interaction between synthesis operators and circuit representations, while a permutation-aware self-supervised task supervises circuit states without requiring step-wise labels. Experimental results demonstrate the effectiveness of our approach across circuits of varying functionalities and scales.

Acknowledgments

This work is suppported by the National Key Research and Development Program of China (No. 2023YFF0725103, 2022YFB2901100), the National Natural Science Foundation of China (No. U22B2038, 62192784, 62404021), Young Elite Scientists Sponsorship Program (No. 2023QNRC001) by CAST, and the Beijing Natural Science Foundation (No. 4244107, QY24216, QY24204).

References

[1] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, et al. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1724–1734.

[2] Animesh Basak Chowdhury et al. 2021. OpenABC-D: A Large-Scale Dataset For Machine Learning Guided Integrated Circuit Synthesis. arXiv:2110.11292 [cs.LG]

[3] Wenji Fang, Yao Lu, Shang Liu, et al. 2023. MasterRTL: A Pre-Synthesis PPA Estimation Framework for Any RTL Design. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9.

[4] Chang Feng et al. 2022. Batch Sequential Black-Box Optimization with Embedding Alignment Cells for Logic Synthesis. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Article 56, 9 pages.

[5] Guyue Huang, Jingbo Hu, Yifan He, et al. 2021. Machine Learning for Electronic Design Automation: A Survey. *ACM Trans. Des. Autom. Electron. Syst.* 26, 5, Article 40 (June 2021), 46 pages.

[6] Dana Lahat et al. 2015. Multimodal data fusion: an overview of methods, challenges, and prospects. *Proc. IEEE* 103, 9 (2015), 1449–1477.

[7] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*. 12888–12900.

[8] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, et al. 2021. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems* 34 (2021), 9694–9705.

[9] Min Li, Sadaf Khan, Zhengyuan Shi, et al. 2022. DeepGate: learning neural representations of logic gates. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*. 667–672.

[10] Hao Liu, Jiarui Feng, Lecheng Kong, et al. 2024. One For All: Towards Training One Graph Model For All Classification Tasks. In *Proceedings of The Twelfth International Conference on Learning Representations (ICLR)*.

[11] Jiawei Liu, Zhiyan Liu, Xun He, et al. 2025. WideGate: Beyond Directed Acyclic Graph Learning in Subcircuit Boundary Prediction. In *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE)*.

[12] Jiawei Liu, Jianwang Zhai, Mingyu Zhao, et al. 2025. PolarGate: Breaking the Functionality Representation Bottleneck of And-Inverter Graph Neural Network. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Article 185, 9 pages.

[13] Yikang Ouyang, Sicheng Li, Dongsheng Zuo, et al. 2023. ASAP: Accurate Synthesis Analysis and Prediction with Multi-Task Learning. In *Proceedings of ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*. 1–6.

[14] Zehua Pei, Fangzhou Liu, Zhuolun He, et al. 2023. AlphaSyn: Logic Synthesis Optimization with Efficient Monte Carlo Tree Search. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9.

[15] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of International Conference on Machine Learning (ICML)*. 8748–8763.

[16] Nan Wu, Jiwon Lee, Yuan Xie, et al. 2022. LOSTIN: Logic Optimization via Spatio-Temporal Information with Hybrid Graph Models. In *Proceedings of 33rd IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 11–18.

[17] Jianyong Yuan, Peiyu Wang, Junjie Ye, et al. 2023. Easyso: Exploration-enhanced reinforcement learning for logic synthesis sequence optimization and a comprehensive RL environment. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9.

[18] Jianan Zhao, Meng Qu, Chaozhuo Li, et al. 2023. Learning on Large-scale Text-attributed Graphs via Variational Inference. In *Proceedings of The Eleventh International Conference on Learning Representations (ICLR)*.

[19] Haisheng Zheng, Zhuolun He, Fangzhou Liu, et al. 2024. LSTP: A Logic Synthesis Timing Predictor. In *29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 728–733.