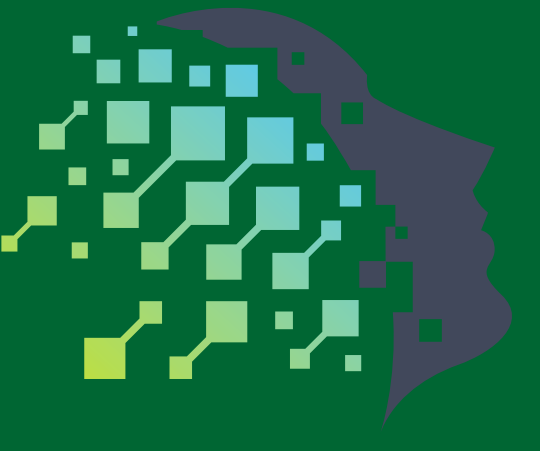




# Circuit Representation Learning with Masked Gate Modeling and Verilog-AIG Alignment

Haoyuan Wu<sup>♠†</sup> Haisheng Zheng<sup>♡†</sup> Yuan Pu<sup>♠♣</sup> Bei Yu<sup>♠</sup>

♠ The Chinese University of Hong Kong ♡ Shanghai Artificial Intelligence Laboratory ♣ ChatEDA Technolgy



## Introduction

### Background:

- Circuits can be formulated as directed acyclic graphs and GNNs can be widely used to learn the characteristics of circuits.
- Self-supervised learning is suitable for circuit representation learning considering that unlabeled circuit data is available and abundant.

### Challenge:

Traditional masked graph modeling paradigm can not be applied directly to circuit representation learning which follows strict logical equivalence. In conventional applications, such as social or molecular graphs, masking nodes can provide a unique solution for reconstruction. However, when gates are masked in a circuit, their reconstruction will admit various solutions.

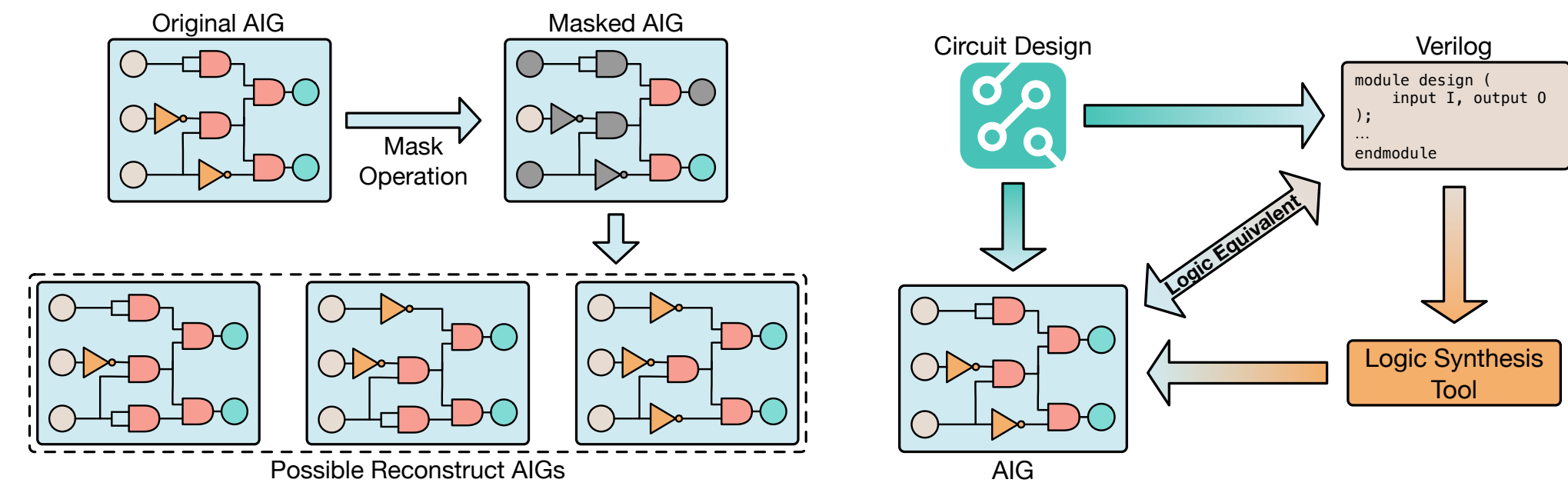


Figure 1. Possible Reconstruct AIGs for Masked AIG. If Circuit Gates are Masked, There are Various Logic-Correct Solutions for Reconstruction.

Figure 2. Logic Equivalence Between Verilog Code and AIG. For a Circuit Design, AIG can be Translated from Verilog Code.

### Contribution:

- Propose masked gate modeling (MGM) for circuit representation learning, enabling GNNs to extract circuit representations with fine-grained structural information.
- Develop the Verilog-AIG alignment (VGA), which employs LLMs as teachers to guide GNNs to extract circuit representations with abstract circuit functions through equivalent AIGs and Verilog codes alignment.
- Conduct extensive evaluations and show superior performance on various logic synthesis tasks including quality of result (QoR) and logic equivalence identification compared to previous state-of-the-art (SOTA) methods.

## AIG Autoencoder

Let  $\mathcal{G} = (\mathcal{V}, \mathcal{A})$  represent an AIG, where  $\mathcal{V}$  denotes the set of  $N$  nodes,  $v_i \in \mathcal{V}$ , categorized into four types: PI, PO, AND, and NOT gates, each labeled by  $c_i \in \mathcal{C}, i \in \{1, 2, 3, 4\}$ .  $\mathcal{A} \in \{0, 1\}^{N \times N}$  is the adjacency matrix, where  $\mathcal{A}_{i,j} = 1$  represents an existing edge from  $v_i$  to  $v_j$ .

For an AIG autoencoder, a GNN encoder, denoted by  $g_E$ , encodes  $\mathcal{G}$  into a latent space representation  $\vec{X} \in \mathbb{R}^{N \times d}$ , where  $d$  represents the dimension of this representation. The encoding process of an AIG can be formulated as:

$$\vec{X} = g_E(\mathcal{V}, \mathcal{A}). \quad (1)$$

Concurrently, a GNN decoder,  $g_D$ , endeavors to reconstruct the AIG  $\mathcal{G}$  from  $\vec{X}$  according to:

$$(\tilde{\vec{X}}, \mathcal{A}) = \tilde{\mathcal{G}} = g_D(\vec{X}, \mathcal{A}), \quad (2)$$

where  $\tilde{\mathcal{G}}$  denotes the reconstructed graph.

## Masked Gate Modeling

We uniformly sample a subset of gates  $\mathcal{V}_{\text{mgm}} \subset \mathcal{V}$  without replacement and replace the remaining nodes with the mask token [MASK], which can be represented by a learnable vector  $\vec{m} \in \mathbb{R}^d$ . Consequently, the masked node representation  $\vec{x}_i \in \vec{X}_{\text{mgm}}$  for each node  $v_i$  is given by:

$$\vec{x}_i = \begin{cases} \vec{x}_i, & \text{if } v_i \in \mathcal{V}_{\text{mgm}}; \\ \vec{m}, & \text{if } v_i \notin \mathcal{V}_{\text{mgm}}. \end{cases} \quad (3)$$

The  $\vec{X}_{\text{mgm}}$  is then fed into the decoder  $g_D$  to reconstruct the  $\mathcal{G}$

## Verilog-AIG Alignment

We uniformly sample a subset of gates  $\mathcal{V}_{\text{vga}} \subset \mathcal{V}$  without replacement and replace the node types of remaining nodes with  $c_m$ , which represents these nodes are masked in the original AIG. Consequently, for  $v_i \in \bar{\mathcal{V}}$  of the masked AIG, the node type  $c_i$  can be defined as:

$$c_i = \begin{cases} c_i, & \text{if } v_i \in \mathcal{V}_{\text{vga}}; \\ c_m, & \text{if } v_i \notin \mathcal{V}_{\text{vga}}. \end{cases} \quad (4)$$

The masked AIG  $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \mathcal{A})$  is fed into the encoder  $g_E$  to generate the encoded masked AIG representation  $\bar{\vec{X}}_{\text{vga}}$ . The reconstruction of  $\mathcal{G}$  from  $\bar{\vec{X}}_{\text{vga}}$  is constrained by equivalent Verilog code to ensure strict logical equivalence.

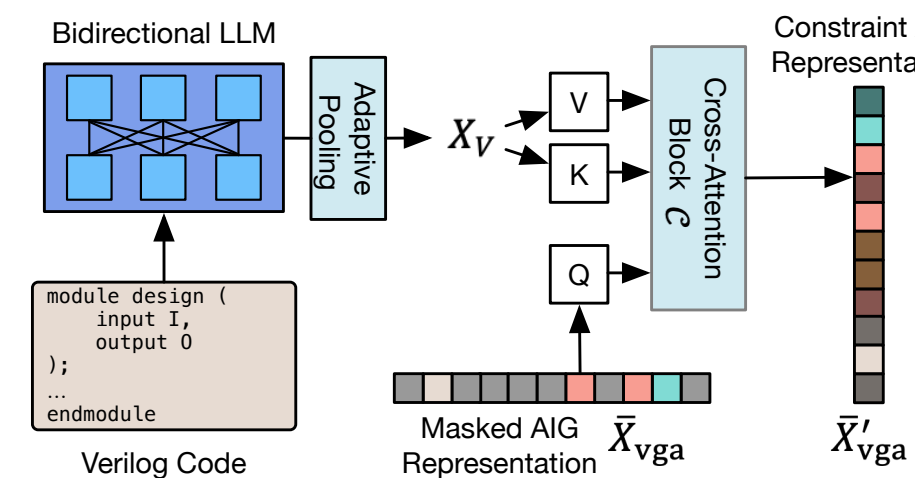


Figure 3. The Constraint Block for VGA.

## AIG Reconstruction

- For gate type prediction,  $\tilde{\vec{X}}$  is transformed by a mapping function  $f_{\text{type}}: \mathbb{R}^d \rightarrow \mathbb{R}^C$  into a categorical probability distribution over  $C$  classes.
- The gate-level degree prediction involves forecasting the in-degree and out-degree of each masked gate within the AIG. Given the reconstructed node representations  $\tilde{\vec{X}}$ , in-degree labels  $\vec{D}^- \in \mathbb{R}^N$ , and out-degree labels  $\vec{D}^+ \in \mathbb{R}^N$ , we utilize mean squared error as the loss function for degree regression tasks.

## Overview of MGPGA

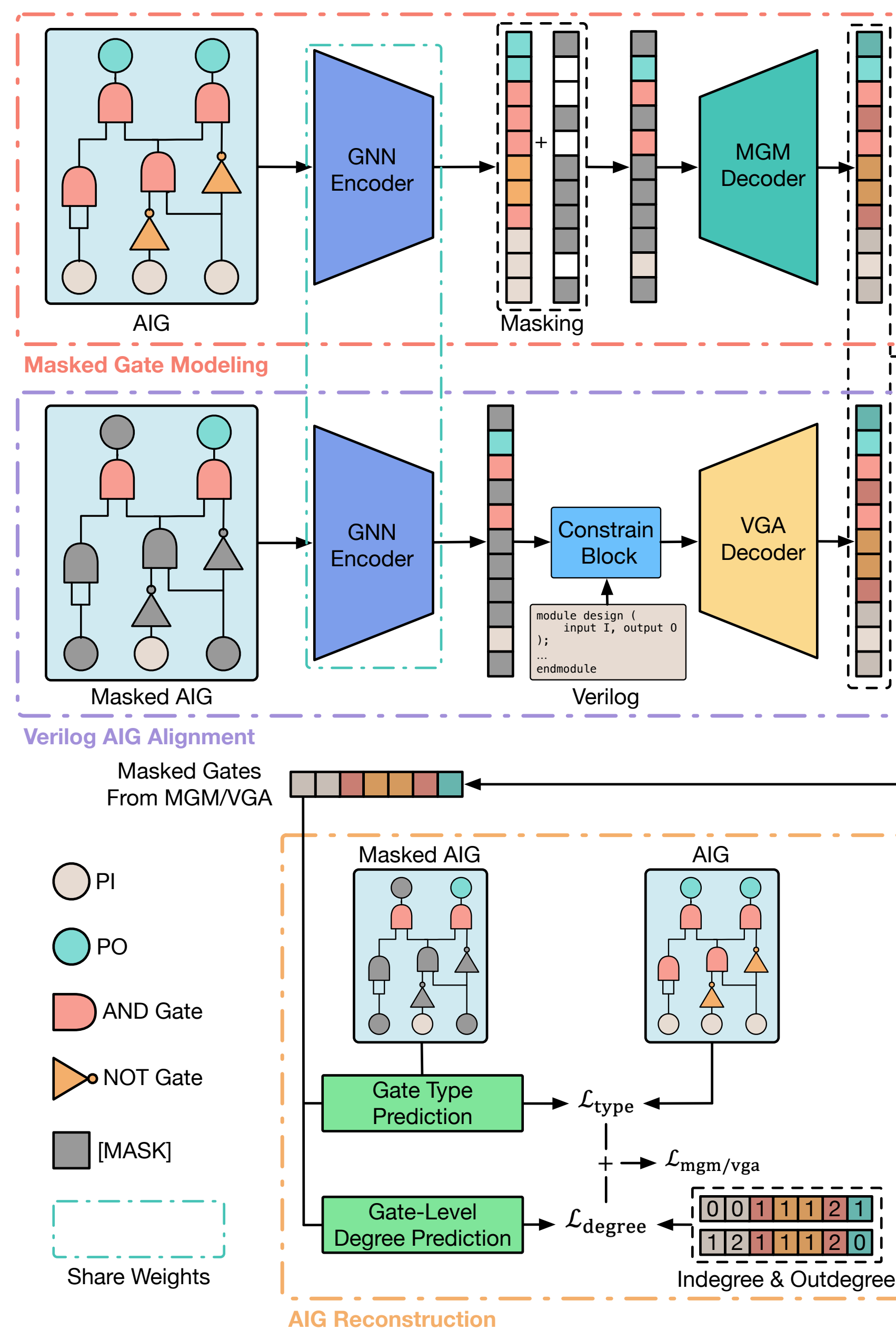


Figure 4. Overview of the MGPGA for Circuit Representation including Masked Gate Modeling and Verilog-AIG Alignment.

## Experiment Settings

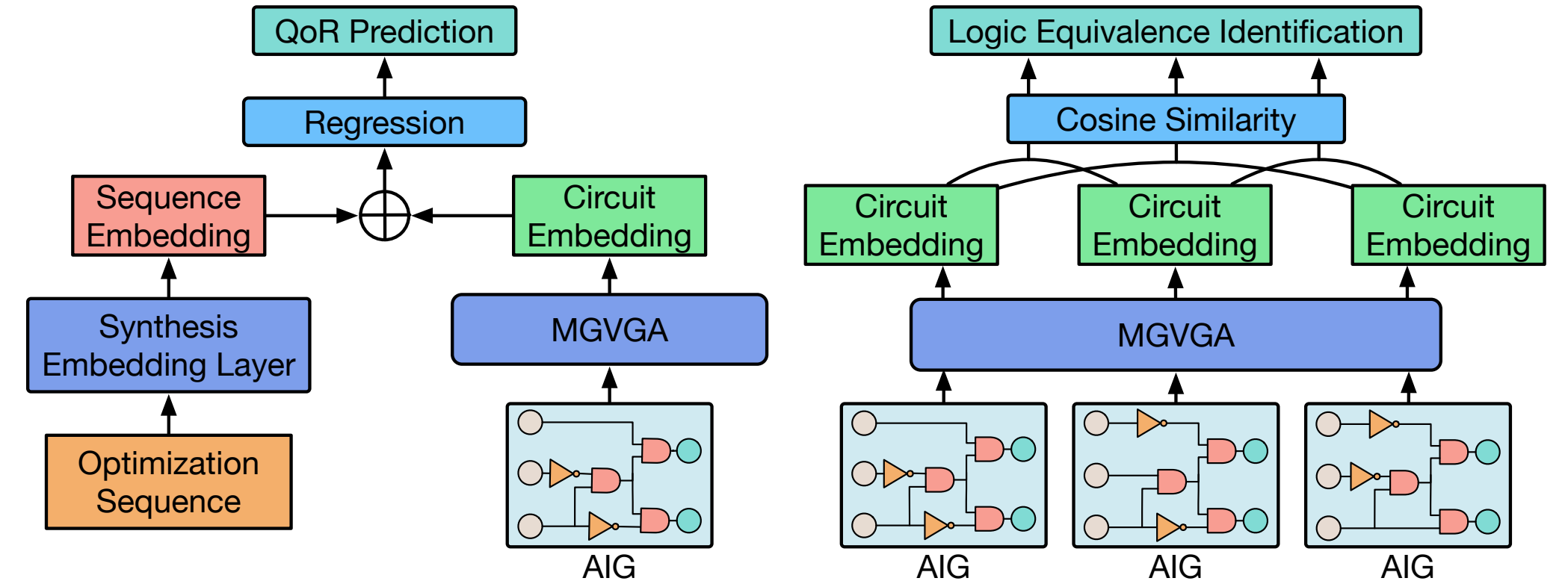


Figure 5. Application of MGPGA in QoR Prediction and Logic Equivalence Identification.

## Main Experiment Results

Table 1. Performance of DeepGate2 and MGPGA on QoR Prediction.

| Design     | # PI | # PO | # Gates | DeepGate2 |          |               |       |       | MGPGA (Ours) |          |               |       |       |
|------------|------|------|---------|-----------|----------|---------------|-------|-------|--------------|----------|---------------|-------|-------|
|            |      |      |         | NDCG@k ↑  | Top-k% ↑ | Commonality ↑ |       |       | NDCG@k ↑     | Top-k% ↑ | Commonality ↑ |       |       |
|            |      |      |         | k=3       | k=5      | k=3           | k=5   | k=10  | k=3          | k=5      | k=3           | k=5   | k=10  |
| bc0        | 21   | 11   | 2784    | 0.331     | 0.395    | 0.244         | 0.227 | 0.280 | 0.444        | 0.560    | 0.222         | 0.213 | 0.320 |
| apex1      | 45   | 45   | 2661    | 0.645     | 0.643    | 0.222         | 0.333 | 0.413 | 0.706        | 0.716    | 0.311         | 0.400 | 0.513 |
| div        | 128  | 128  | 101698  | -0.063    | 0.029    | 0.000         | 0.027 | 0.133 | -0.060       | -0.060   | 0.000         | 0.013 | 0.093 |
| k2         | 45   | 45   | 4075    | -0.060    | 0.040    | 0.022         | 0.040 | 0.080 | 0.902        | 0.873    | 0.267         | 0.320 | 0.400 |
| i10        | 257  | 224  | 3618    | -0.133    | -0.080   | 0.000         | 0.000 | 0.027 | 0.620        | 0.607    | 0.289         | 0.307 | 0.353 |
| mainpla    | 26   | 49   | 9441    | 0.674     | 0.629    | 0.267         | 0.293 | 0.360 | 0.594        | 0.598    | 0.200         | 0.187 | 0.233 |
| or1200_cpu | 2343 | 2072 | 56570   | 0.498     | 0.485    | 0.178         | 0.267 | 0.407 | 0.617        | 0.613    | 0.222         | 0.253 | 0.367 |
| picorv32   | 1631 | 1601 | 25143   | 0.563     | 0.406    | 0.111         | 0.173 | 0.186 | 0.440        | 0.457    | 0.066         | 0.160 | 0.180 |
| Rocket     | 4413 | 4187 | 96507   | 0.578     | 0.543    | 0.111         | 0.186 | 0.300 | 0.557        | 0.607    | 0.355         | 0.413 | 0.467 |
| sqrt       | 128  | 64   | 40920   | 0.304     | 0.153    | 0.000         | 0.027 | 0.080 | 0.577        | 0.401    | 0.000         | 0.040 | 0.080 |
| Average    |      |      |         | 0.334     | 0.324    | 0.116         | 0.157 | 0.226 | 0.540        | 0.537    | 0.193         | 0.231 | 0.301 |

Table 2. Performance of DeepGate2 and MGPGA on Logic Equivalence Identification.

| Design     | DeepGate2 |        |          |       | MGPGA (Ours) |        |          |       |
|------------|-----------|--------|----------|-------|--------------|--------|----------|-------|
|            | Precision | Recall | F1-Score | AUC   | Precision    | Recall | F1-Score | AUC   |
| bc0        | 0.199     | 0.930  | 0.327    | 0.813 | 0.274        | 0.715  | 0.396    | 0.817 |
| apex1      | 0.133     | 0.680  | 0.223    | 0.601 | 0.273        | 0.710  | 0.394    | 0.826 |
| div        | 0.203     | 0.980  | 0.337    | 0.814 | 0.197        | 0.670  | 0.305    | 0.725 |
| k2         | 0.171     | 0.720  | 0.276    | 0.695 | 0.336        | 0.920  | 0.492    | 0.919 |
| i10        | 0.414     | 0.940  | 0.575    | 0.918 | 0.699        | 0.950  | 0.805    | 0.985 |
| mainpla    | 0.178     | 0.790  | 0.290    | 0.732 | 0.167        | 0.900  | 0.281    | 0.746 |
| or1200_cpu | 0.451     | 0.790  | 0.575    | 0.823 | 0.356        | 0.950  | 0.518    | 0.929 |
| picorv32   | 0.448     | 0.870  | 0.592    | 0.918 | 0.440        | 0.960  | 0.604    | 0.941 |
| Rocket     | 0.346     | 0.930  | 0.504    | 0.892 | 0.388        | 1.000  | 0.559    | 0.952 |
| sqrt       | 0.189     | 0.740  | 0.302    | 0.721 | 0.199        | 0.720  | 0.312    | 0.770 |
| Average    | 0.295     | 0.841  | 0.424    | 0.804 | 0.336        | 0.848  | 0.470    | 0.862 |

## Analysis of Masking Ratio

Table 3. Performance of MGPGA on QoR Prediction and Logic Equivalence Identification with Different Masking Ratios of MGM and VGA.

| Masking Ratio |     | QoR Prediction |          |               |       |       | Logic Equivalence Identification |        |          |       |
|---------------|-----|----------------|----------|---------------|-------|-------|----------------------------------|--------|----------|-------|
| MGM           | VGA | NDCG@k ↑       | Top-k% ↑ | Commonality ↑ |       |       | Precision                        | Recall | F1-Score | AUC   |
|               |     | k=3            | k=5      | k=3           | k=5   | k=10  |                                  |        |          |       |
| 0.3           | 0.3 | 0.517          | 0.505    | 0.158         | 0.199 | 0.272 | 0.300                            | 0.844  | 0.430    | 0.846 |
| 0.3           | 0.5 | 0.540          | 0.537    | 0.193         | 0.231 | 0.301 | 0.336                            | 0.848  | 0.470    | 0.862 |
| 0.3           | 0.7 | 0.498          | 0.514    | 0.178         | 0.223 | 0.299 | 0.316                            | 0.823  | 0.441    | 0.820 |
| 0.5           | 0.3 | 0.439          | 0.445    | 0.149         | 0.187 | 0.271 | 0.274                            | 0.821  | 0.402    | 0.821 |
| 0.5           | 0.5 | 0.438          | 0.470    | 0.169         | 0.204 | 0.288 | 0.331                            | 0.829  | 0.450    | 0.836 |
| 0.5           | 0.7 | 0.385          | 0.415    | 0.140         | 0.168 | 0.247 | 0.304                            | 0.833  | 0.433    | 0.817 |
| 0.7           | 0.3 | 0.347          | 0.367    | 0.127         | 0.160 | 0.242 | 0.292                            | 0.871  | 0.421    | 0.828 |
| 0.7           | 0.5 | 0.400          | 0.366    | 0.111         | 0.175 | 0.228 | 0.313                            | 0.823  | 0.433    | 0.832 |
| 0.7           | 0.7 | 0.366          | 0.359    | 0.138         | 0.169 | 0.226 | 0.305                            | 0.794  | 0.425    | 0.822 |

## Generalization on Various GNNs

Table 4. Generalization on Various GNNs, including GraphSAGE and Graph Transformer.

| GNNs              | QoR Prediction |          |               |       |       | Logic Equivalence Identification |        |          |       |
|-------------------|----------------|----------|---------------|-------|-------|----------------------------------|--------|----------|-------|
|                   | NDCG@k ↑       | Top-k% ↑ | Commonality ↑ |       |       | Precision                        | Recall | F1-Score | AUC   |
|                   | k=3            | k=5      | k=3           | k=5   | k=10  |                                  |        |          |       |
| DeepGate2         | 0.334          | 0.324    | 0.116         | 0.157 | 0.226 | 0.295                            | 0.841  | 0.424    | 0.804 |
| GraphSAGE         | 0.469          | 0.479    | 0.153         | 0.224 | 0.314 | 0.329                            | 0.811  | 0.455    | 0.841 |
| Graph Transformer | 0.452          | 0.470    | 0.154         | 0.212 | 0.311 | 0.324                            | 0.789  | 0.450    | 0.831 |
| DeepGCN (Ours)    | 0.540          | 0.537    | 0.193         | 0.231 | 0.301 | 0.336                            | 0.848  | 0.470    | 0.862 |