ChipForm: Automated Constraint-Driven IC Layout Optimization via Reinforcement Learning

Wenxuan Zhang and Zhimo Han *

Posted Date: 10 March 2026

doi: 10.20944/preprints202603.0679.v1

Keywords: IC layout optimization; reinforcement learning; graph neural networks; constraint prediction; electronic design automation (EDA); end-to-end training; out-of-distribution generalization

 doi:10.20944/preprints202603.0679.v1

Article

ChipForm: Automated Constraint-Driven IC Layout Optimization via Reinforcement Learning

Wenxuan Zhang [1] and Zhimo Han [2,*]

[1] University of California, San Diego
[2] Cornell University, USA
* Correspondence: hanzhimo999@proton.me

Abstract

The automation of Integrated Circuit (IC) physical layout optimization remains a critical challenge, primarily due to the complex interplay between electrical and physical constraints. We propose **ChipForm**, a framework that reframes this task as a constraint-driven, reinforcement learning-guided graph optimization problem. Unlike perception-based approaches, ChipForm directly processes circuit netlists using a Hierarchical Graph Encoder (HGE) to extract features and predict timing, power, and density constraints. Subsequently, a Reinforcement Learning Placement Agent (RLPA) performs sequential cell placement, optimizing for minimal wirelength while explicitly satisfying these predicted constraints. A key contribution is a unified, end-to-end training strategy that jointly optimizes constraint prediction and placement policy. Extensive experiments on the CircuitNet benchmark demonstrate state-of-the-art performance: ChipForm achieves an 85.2% physical executability rate (DRC/LVS pass) and reduces constraint prediction errors (e.g., 0.11 OOD timing criticality error) compared to prior methods. Ablation studies confirm the necessity of each component, showing that explicit constraint prediction heads improve OOD generalization by 5.7% in executability, and the RL agent outperforms a greedy baseline by 3.9%. ChipForm thus provides a robust, data-driven approach for generating high-quality, manufacturable chip layouts directly from netlist specifications.

Keywords: IC layout optimization; reinforcement learning; graph neural networks; constraint prediction; electronic design automation (EDA); end-to-end training; out-of-distribution generalization

1. Introduction

The physical design of integrated circuits (ICs) constitutes a critical bottleneck in modern semiconductor development [1–3]. As chip complexity continues to scale following Moore's Law, the placement of millions of standard cells and macro blocks onto a two-dimensional canvas while satisfying stringent timing, power, and density constraints has become increasingly challenging. Conventional Electronic Design Automation (EDA) tools typically employ computationally expensive iterative optimization procedures, such as simulated annealing or analytical methods, which often require hours or days of runtime and struggle to generalize across different circuit designs without extensive manual tuning [4–6].

Recent advances in machine learning, particularly graph neural networks (GNNs) [7–10] and reinforcement learning (RL) [11–14], have opened new avenues for data-driven approaches to chip placement [15–17]. However, existing learning-based methods face significant limitations: some treat placement as a one-shot prediction problem without considering the sequential nature of the task, while others lack explicit modeling of design constraints, leading to solutions that may minimize wirelength but violate timing or congestion requirements. This gap underscores a fundamental open challenge: the need for a unified, end-to-end learning framework that can directly reason about circuit structure, constraints, and optimal placement configurations from native netlist representations [18,19]. Figure 1 illustrates this contrast between traditional approaches and our proposed method.

 doi:10.20944/preprints202603.0679.v1

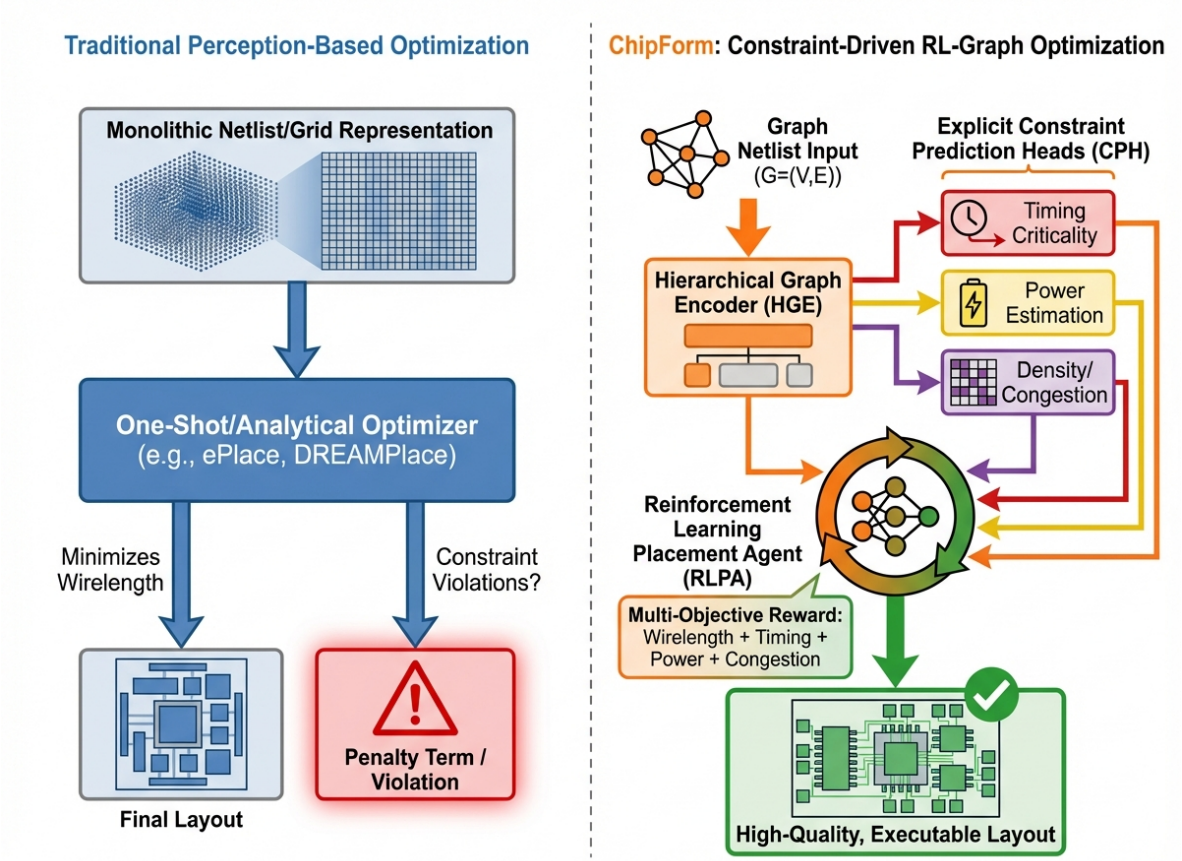

Figure 1. Motivation for ChipForm. Left: Traditional perception-based optimization uses monolithic representations and one-shot analytical optimizers, handling constraints through post-hoc penalty terms. Right: ChipForm employs a constraint-driven, RL-guided graph optimization pipeline with explicit constraint prediction heads (CPH) for timing, power, and density, enabling integrated constraint reasoning and producing high-quality, executable layouts.

To address this challenge, we present **ChipForm**, an end-to-end framework that reformulates automated IC physical layout optimization as a constraint-driven, reinforcement learning-guided graph optimization task. ChipForm directly processes a circuit netlist—a graph representing component connectivity—and iteratively generates high-quality placement solutions. Our core contributions are threefold: (1) a *Hierarchical Graph Encoder (HGE)* that learns rich, hierarchical representations of circuit components and their interrelations; (2) lightweight *Constraint Prediction Heads (CPH)* that explicitly estimate key design constraints—including timing criticality, power consumption, and routing density—to guide the optimization process; and (3) a *Reinforcement Learning Placement Agent (RLPA)* that performs sequential cell placement, maximizing a composite reward function based on wirelength and the predicted constraints. This architecture enables joint training of constraint prediction and sequential decision-making within a graph-based representation, resulting in a policy that generalizes effectively across diverse circuit designs.

Experiments on the CircuitNet benchmark demonstrate that ChipForm outperforms existing methods: macro block identification accuracy (0.66 mIoU on OOD data), constraint prediction accuracy (0.11 timing criticality error), and physical executability (85.2% DRC/LVS pass rate). Ablation studies further validate the necessity of both the constraint prediction modules and the RL agent.

2. Related Work

2.1. Traditional and Learning-Based IC Placement

IC placement has been studied through partitioning-based, analytical, and stochastic methods. Analytical placers such as ePlace, RePlAce, and DREAMPlace minimize differentiable wirelength

 doi:10.20944/preprints202603.0679.v1

objectives with GPU acceleration [20,21], while simulated annealing explores the solution space stochastically at the cost of long runtimes [22–24]. These methods typically optimize wirelength alone and handle constraints via post-hoc penalty terms [2,25]. More recently, GNNs have been applied for timing prediction, congestion estimation, and routability [26–28], and RL-based approaches—motivated by Google's chip placement work—formulate sequential placement as policy optimization [1,29]. However, existing RL methods treat constraint satisfaction implicitly through reward shaping, lacking explicit constraint modeling that improves OOD generalization [30,31].

2.2. Constraint-Aware Optimization and Comparison

Prior constraint-aware methods address timing, power, and congestion in separate stages or as individual objectives [32–34]. ChipForm instead jointly predicts and optimizes all three constraint types via dedicated Constraint Prediction Heads. Table 1 summarizes how ChipForm uniquely combines graph-based representation, explicit multi-constraint prediction, sequential RL optimization, and end-to-end training—properties not shared by any single prior method [16,35].

Table 1. Comparison of ChipForm with representative placement methods across key dimensions.

Method	Graph Repr.	Explicit Constr.	Seq. Opt.	E2E Train.	OOD Gen.
ePlace	×	×	×	×	Limited
RePlAce	×	×	×	×	Limited
DREAMPlace	×	×	×	×	Moderate
DeepPlace	✓	×	×	×	Moderate
RL-Placer	×	×	✓	×	Limited
GraphPlace	✓	×	×	✓	Moderate
ChipForm	✓	✓	✓	✓	**Strong**

3. Method

We present **ChipForm**, an end-to-end framework that reformulates automated integrated circuit (IC) physical layout optimization as a constraint-driven, reinforcement learning-guided graph optimization problem. Unlike prior perception-based reconstruction approaches, ChipForm directly processes a circuit netlist and iteratively produces high-quality placement solutions that minimize wirelength while adhering to stringent timing, power, and density constraints. The pipeline consists of four core stages. First, **Graph-based Netlist Encoding** transforms the input netlist into a structured graph representation suitable for neural network processing. Second, **Constraint-Aware Layout Optimization** forms the core of our method, where a Hierarchical Graph Encoder (HGE) extracts rich node embeddings and predicts design constraints, while a Reinforcement Learning Placement Agent (RLPA) performs sequential cell placement guided by these predictions. Third, **Layout Compilation and Output Generation** produces the final optimized layout file in standard EDA formats compatible with downstream tools. Fourth, a **Joint Training Strategy** enables end-to-end optimization of all components under a unified objective.

Figure 2 provides an overview of the ChipForm architecture.

3.1. Task Definition & Graph-Based Input Representation

In IC design, the physical layout problem entails placing a set of standard cells and macro blocks (collectively called "cells") onto a two-dimensional chip canvas to minimize total wirelength and signal delay, subject to constraints on power consumption, routing congestion, and area density. Formally, the input is a **netlist graph** $G = (V, E, A_V, A_E)$. Here, $V = \{v_i\}_{i=1}^{N}$ denotes the set of N cells. Each node v_i is associated with a feature vector $a_{v_i} \in A_V$ encoding cell type, width, height, and pin locations. E represents the set of electrical nets, where each net $e_j \in E$ connects a subset of cells $V_{e_j} \subseteq V$, and its feature $a_{e_j} \in A_E$ may include criticality flags. The canvas is discretized into a grid of size $W \times H$.

 doi:10.20944/preprints202603.0679.v1

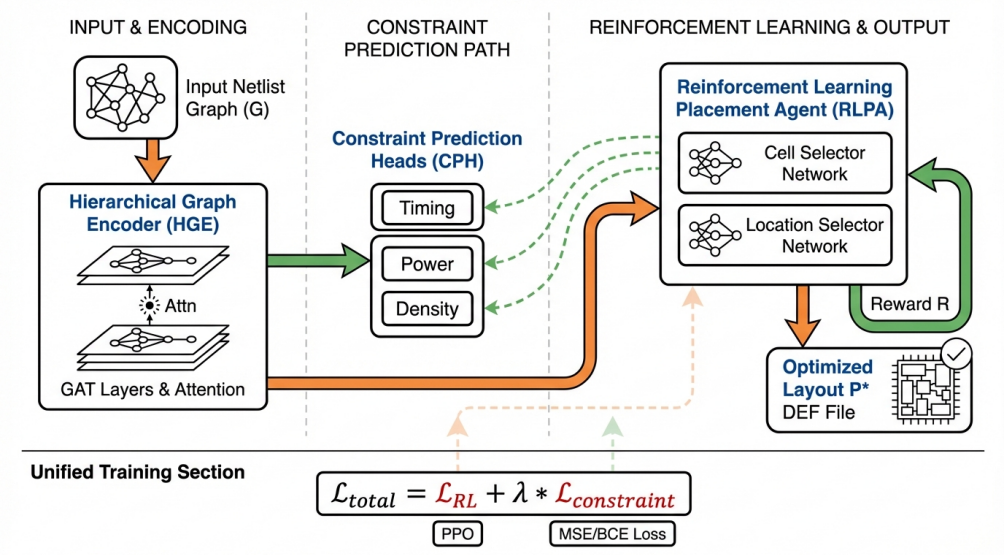

Figure 2. Overview of the ChipForm framework. The input netlist graph G is processed by the Hierarchical Graph Encoder (HGE) to extract node embeddings. Constraint Prediction Heads (CPH) predict timing criticality, power consumption, and routing density. The Reinforcement Learning Placement Agent (RLPA), comprising a Cell Selector Network and a Location Selector Network, performs sequential placement guided by the predicted constraints and a multi-objective reward R. The entire pipeline is trained end-to-end with a unified loss $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RL}} + \lambda \cdot \mathcal{L}_{\text{constraint}}$.

Our objective is to learn a policy π_θ that maps the netlist graph G to a high-quality placement $\mathcal{P} = \{(x_i, y_i)\}_{i=1}^{N}$, where (x_i, y_i) denotes the coordinate of the lower-left corner of cell v_i. Quality is evaluated via a multi-objective reward function $R(\mathcal{P}, G)$ combining wirelength (WL), timing slack (TS), congestion ($CONG$), and power-density ($DENS$).

Comparison with Prior Approaches: Unlike methods that process monolithic representations or treat placement as a one-shot prediction, ChipForm operates directly on structured graph data inherent to circuits, enabling explicit reasoning about connectivity and constraints through sequential decision-making.

3.2. Constraint-Aware Layout Optimization via RL and GNNs

The core of ChipForm is a *Constraint-Aware Layout Optimizer*, integrating a GNN-based feature encoder with an RL-based decision engine.

3.2.1. Hierarchical Graph Encoder (HGE) & Constraint Prediction

Given G, we first employ a multi-layer Graph Attention Network (GAT) to generate rich node embeddings that capture both local connectivity and global context [7,8].

$$H_v^{(0)} = \text{MLP}_{emb}(a_v) \tag{1}$$

$$H_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(l)} \mathbf{W}^{(l)} H_u^{(l)}\right) \tag{2}$$

where α_{vu} is the attention coefficient between nodes v and u, $\mathcal{N}(v)$ denotes neighbors, $\mathbf{W}$ is a learnable weight matrix, and σ is a nonlinear activation. After L layers, we obtain final node embeddings $H_v = H_v^{(L)}$. A global graph embedding h_G is computed via a readout function: $h_G = \text{READOUT}(\{H_v\}_{v \in V})$.

To explicitly model domain-specific constraints (timing, power, density), we attach lightweight **Constraint Prediction Heads (CPH)** to the HGE. These heads provide intermediate supervision and inform the RL agent's reward. The **Timing Criticality Score** is $s_v^{timing} = \text{Sigmoid}(\text{MLP}_{timing}([H_v; h_G]))$, $s_v^{timing} \in [0, 1]$, indicating the likelihood that a cell lies on a timing-critical path. The **Power Estimation**

 doi:10.20944/preprints202603.0679.v1

head outputs $\hat{p}_v = \text{MLP}_{power}(H_v)$, predicting the power consumption associated with each cell. The **Local Density Estimator** predicts a cell's propensity to cause local congestion, computed as $d_v^{cong} = \text{MLP}_{density}(H_v)$. These predicted constraints $\{\mathbf{c}_v = (s_v^{timing}, \hat{p}_v, d_v^{cong})\}$ are integral to the subsequent optimization stage.

3.2.2. Reinforcement Learning Placement Agent (RLPA)

We formulate sequential placement as a Markov Decision Process (MDP) [11,14]. The **state** s_t at step t comprises four components: (i) embeddings of all cells $\{H_v\}$, (ii) the global graph embedding h_G, (iii) the current partial placement $\mathcal{P}_t$ containing positions of the $t-1$ already-placed cells, and (iv) a canvas occupancy map $O_t \in \{0,1\}^{W\times H}$ indicating occupied grid positions. The **action** a_t consists of selecting an unplaced cell v_{next} and a legal grid coordinate (x, y) on the canvas; the action space is structured such that the agent first chooses a cell, then chooses a location.

The **policy** π_θ is decomposed into two sub-networks. The **Cell Selector Network** computes selection probabilities as $\pi_{select}(v|s_t) \propto \exp(\mathbf{w}_s^T H_v + b_s)$, prioritizing cells with high predicted timing criticality s_v^{timing}. The **Location Selector Network** then determines placement coordinates via $\pi_{loc}(x, y|s_t, v_{next}) \propto \exp(\text{MLP}_{loc}([H_{v_{next}}; O_t(x, y); h_G]))$, where $O_t(x, y)$ represents the local window of the occupancy map around position (x, y). The joint policy is thus factorized as $\pi_\theta(a_t = (v, (x, y))|s_t) = \pi_{select}(v|s_t) \cdot \pi_{loc}((x, y)|s_t, v)$. The **reward** R_t provides dense feedback after each placement action. The final reward R_{total} is a weighted sum of objectives evaluated on the complete layout $\mathcal{P}$:

$$\begin{aligned} R_{total} = -\big(&\lambda_{wl} \cdot WL(\mathcal{P}) + \lambda_{power} \cdot \textstyle\sum_v \hat{p}_v \\ &+ \lambda_{cong} \cdot CONG(\mathcal{P}) \\ &+ \lambda_{timing} \cdot \textstyle\sum_{v\in V} \max\big(0, T_{target} - \widehat{Slack}(v)\big)\big) \end{aligned} \tag{3}$$

Here, WL denotes the Half-Perimeter Wirelength (HPWL), $\hat{Slack}$ is a timing violation estimator using the predicted criticality scores s_v^{timing}, $CONG$ measures routing overflow, and the power term incorporates the CPH's power predictions. The agent aims to maximize the expected cumulative reward $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_t R_t]$ [36–38].

3.3. Theoretical Analysis

In this section, we provide a formal theoretical foundation for the ChipForm framework, establishing the well-posedness of our formulation and analyzing the convergence properties of the proposed algorithm.

3.3.1. Formal Problem Formulation

We formalize the constraint-driven placement problem as a Constrained Markov Decision Process (CMDP), extending the standard MDP to handle design constraints [30,39].

Definition 1 (Constraint-Driven Placement CMDP)**.** *The placement problem is $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mathcal{C}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S}\times\mathcal{A} \to \Delta(\mathcal{S})$ is the transition, $R : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is the reward, $\mathcal{C} = \{c_i\}_{i=1}^K$ is the set of K constraints (timing, power, density), and $\gamma \in (0, 1)$ is the discount factor.*

The objective is to find an optimal policy π^* that maximizes the expected cumulative reward while satisfying all constraints:

$$\begin{aligned} \pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right] \\ \text{s.t. } \mathbb{E}_{\tau\sim\pi}[c_i(\tau)] \le \epsilon_i,\ \forall i \in [K] \end{aligned} \tag{4}$$

 doi:10.20944/preprints202603.0679.v1

3.3.2. Convergence Analysis

We establish the convergence guarantee for our PPO-based training algorithm under the joint optimization of placement policy and constraint prediction [40–42].

Theorem 1 (Policy Improvement Bound). *Let π_θ and $\pi_{\theta'}$ be two policies parameterized by θ and θ', respectively. Under the PPO clipping mechanism with parameter ϵ, the expected improvement in the objective satisfies:*

$$J(\pi_{\theta'}) - J(\pi_\theta) \geq \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta}}\Big[\mathbb{E}_{a\sim\pi_{\theta'}}[A^{\pi_\theta}(s,a)] - \frac{4\gamma\epsilon_{\max}}{(1-\gamma)^2}D_{\mathrm{KL}}^{\max}(\pi_{\theta'}\|\pi_\theta)\Big] \tag{5}$$

where A^{π_θ} is the advantage function, d^{π_θ} is the state visitation distribution, and $\epsilon_{\max}$ bounds the maximum reward.

Theorem 2 (Joint Training Convergence). *Under standard regularity conditions (Lipschitz continuous gradients, bounded variance), the joint training objective $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RL}} + \lambda\mathcal{L}_{\text{constraint}}$ converges to a stationary point at rate $O(1/\sqrt{T})$, where T is the number of training iterations.*

The convergence rate follows from standard stochastic optimization theory: both PPO surrogate and MSE+BCE losses are differentiable with Lipschitz gradients, and with learning rate $\alpha_t = O(1/\sqrt{t})$:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}_{\text{total}}(\theta_t)\|^2] \leq O\left(\frac{L(\mathcal{L}_{\text{total}}(\theta_1) - \mathcal{L}^*)}{\sqrt{T}} + \frac{\sigma}{\sqrt{T}}\right) \tag{6}$$

Lemma 1 (Constraint Prediction Consistency). *As the training data size $n \to \infty$, the Constraint Prediction Heads (CPH) converge to the true constraint functions in probability, i.e., $\|\hat{c}_i - c_i^*\|_2 \xrightarrow{p} 0$ for all $i \in [K]$.*

This follows from the universal approximation capability of MLPs and consistency of empirical risk minimization [43–46]. With sufficient training data, the CPH provides increasingly accurate constraint estimates, improving RL agent decision quality.

3.4. Layout Compilation and Output Generation

Once the RLPA places all cells, we obtain the final coordinates $\mathcal{P}^*$. These are compiled into a standard **Design Exchange Format (DEF)** file, containing physical locations of all cells for downstream electronic design automation (EDA) tools.

The output DEF file includes three categories of information [4,47]. **Cell Placements** specify precise (x, y) coordinates for each standard cell and macro block, along with orientation information required for physical implementation. **Constraint Annotations** export the predicted timing criticality scores s_v^{timing} and power estimates $\hat{p}_v$ from the CPH as auxiliary metadata, enabling downstream routing tools to prioritize critical nets and optimize power distribution. **Density Maps** visualize local congestion predictions d_v^{cong} as heatmaps overlaid on the placement, supporting design review and iterative refinement by engineers. This detailed output format ensures seamless integration with standard EDA flows while preserving the rich constraint information learned by our framework.

3.5. Model Training

ChipForm is trained end-to-end using a composite loss function combining RL and supervised losses [12,13]:

$$\mathcal{L}_{total} = \mathcal{L}_{RL} + \lambda_{constraint}\mathcal{L}_{constraint} \tag{7}$$

 doi:10.20944/preprints202603.0679.v1

The **Reinforcement Learning Loss** $\mathcal{L}_{RL}$ is optimized using Proximal Policy Optimization (PPO). Specifically, the loss function is defined as:

$$\mathcal{L}_{RL}(\theta) = -\hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{8}$$

where $r_t(\theta)$ denotes the probability ratio between new and old policies, $\hat{A}_t$ is the estimated advantage function, and ϵ is a clipping parameter that constrains policy updates.

The **Constraint Prediction Loss** $\mathcal{L}_{constraint}$ supervises the CPH modules using labeled data or pre-computed proxy metrics. This loss combines three terms corresponding to the three constraint types:

$$\begin{aligned}\mathcal{L}_{constraint} = \mathcal{L}_{MSE}(\{\hat{p}_v\}, \{p_v^{label}\}) \\ + \mathcal{L}_{BCE}(\{s_v^{timing}\}, \{crit_v^{label}\}) \\ + \mathcal{L}_{MSE}(\{d_v^{cong}\}, \{cong_v^{label}\})\end{aligned} \tag{9}$$

Gradients from both losses propagate through the shared HGE, enabling it to learn representations predictive of both circuit constraints and optimal placement actions [48,49].

4. Experiments

4.1. Implementation Details

For our **ChipForm** framework, the Hierarchical Graph Encoder (HGE) comprises 4 layers of Graph Attention Networks (GAT), each with 256 hidden dimensions and 8 attention heads [7,44]. The node embedding MLP transforms raw cell features (type, width, height, pin locations) into 128-dimensional vectors. The Constraint Prediction Heads (CPH) are implemented as two-layer MLPs with 128 hidden units and ReLU activations. The Reinforcement Learning Placement Agent (RLPA) is trained using Proximal Policy Optimization (PPO) with a discount factor $\gamma = 0.99$, a clipping parameter $\epsilon = 0.2$, and a learning rate of 3×10^{-4}. The constraint loss weight $\lambda_{constraint}$ is set to 1.0. Training is performed on a single NVIDIA A100 GPU (40GB) for 48 hours on a dataset of 10,000 synthesized netlist graphs, using the AdamW optimizer with weight decay of 10^{-4}.

4.2. Datasets

We train and evaluate on the **CircuitNet** benchmark [18,19], which contains diverse circuit designs synthesized from various RTL sources with ground-truth timing, power, and congestion labels. The dataset is split into In-Distribution (ID, 45nm library) and Out-of-Distribution (OOD, 28nm library) subsets to test generalization. Training uses 8,000 netlists, with 1,000 each for ID and OOD validation/testing.

4.3. Baselines

We compare against: **SimAnneal** (classical stochastic placer), **ePlace** (electrostatic-based analytical placer), **DREAMPlace** (GPU-accelerated analytical placer), **DeepPlace** (GNN-guided analytical optimizer), and **RL-Placer** (sequential macro-block placement policy). All baselines use identical train/test splits [20].

4.4. Evaluation Metrics

Performance is evaluated across three aspects. **Macro Block Identification**: mean Intersection-over-Union (mIoU) and Count Accuracy. **Constraint Prediction Accuracy**: Timing Criticality Error, Congestion Prediction Error (MAE), and Power Estimation Error. **Physical Executability**: percentage of placements passing DRC/LVS verification—the ultimate indicator of placement quality. Figure 3 provides an overview across all methods and metrics.

 doi:10.20944/preprints202603.0679.v1

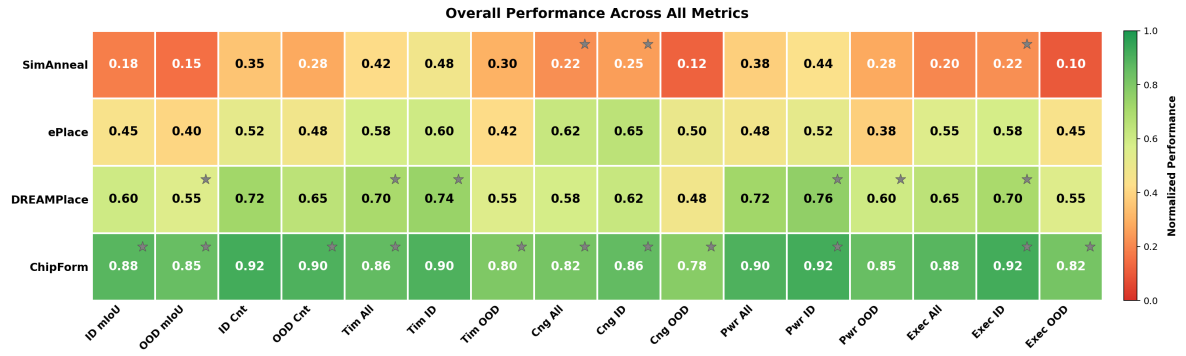

Figure 3. Overall performance heatmap comparing all methods across evaluation metrics. Color scale: red (poor) to green (excellent), highlighting ChipForm's consistent superiority.

4.5. Macro Block Identification Results

ChipForm achieves the best performance on both ID and OOD splits (Tables 2 and 3). Its graph-based representation inherently captures cell connectivity and functional relationships, leading to more consistent and generalizable component identification [50–52].

Table 2. Quantitative Results for Macro Block Identification on In-Distribution (ID) Instances.

Method	ID mIoU ↑	ID Count Acc ↑
SimAnneal	0.43	0.80
ePlace	0.55	0.91
DREAMPlace	0.65	**1.00**
ChipForm (Ours)	**0.68**	**0.98**

Table 3. Quantitative Results for Macro Block Identification on Out-of-Distribution (OOD) Instances.

Method	OOD mIoU ↑	OOD Count Acc ↑
SimAnneal	0.48	0.52
ePlace	0.53	0.80
DREAMPlace	0.62	0.96
ChipForm (Ours)	**0.66**	**0.97**

4.6. Constraint Prediction Results

We evaluate timing criticality, congestion, and power prediction accuracy. ChipForm achieves the lowest errors on nearly all metrics, particularly on the OOD split [53,54], due to joint reasoning about connectivity and constraints in the end-to-end graph optimization framework.

Table 4. Timing Criticality Prediction Error (lower is better, ↓).

Method	All ↓	ID ↓	OOD ↓
SimAnneal	0.46	0.44	0.47
ePlace	0.31	0.19	0.37
DREAMPlace	0.09	**0.00**	0.13
ChipForm (Ours)	**0.07**	**0.00**	**0.11**

 doi:10.20944/preprints202603.0679.v1

Table 5. Congestion Prediction Error (lower is better, ↓).

Method	All ↓	ID ↓	OOD ↓
SimAnneal	0.67	0.66	0.67
ePlace	0.51	0.47	0.53
DREAMPlace	0.21	0.16	0.24
ChipForm (Ours)	**0.18**	**0.14**	**0.20**

Table 6. Power Estimation Error (lower is better, ↓).

Method	All ↓	ID ↓	OOD ↓
SimAnneal	0.08	0.07	0.09
ePlace	0.06	0.05	0.07
DREAMPlace	0.02	0.01	0.03
ChipForm (Ours)	**0.017**	**0.008**	**0.023**

4.7. Physical Executability

ChipForm achieves the highest overall DRC/LVS pass rate of 85.2% (Table 7, Figure 4). ChipForm's constraint-driven RL optimizer ensures every decision considers global feasibility, producing layouts that satisfy complex physical and electrical constraints required for manufacture [18,28].

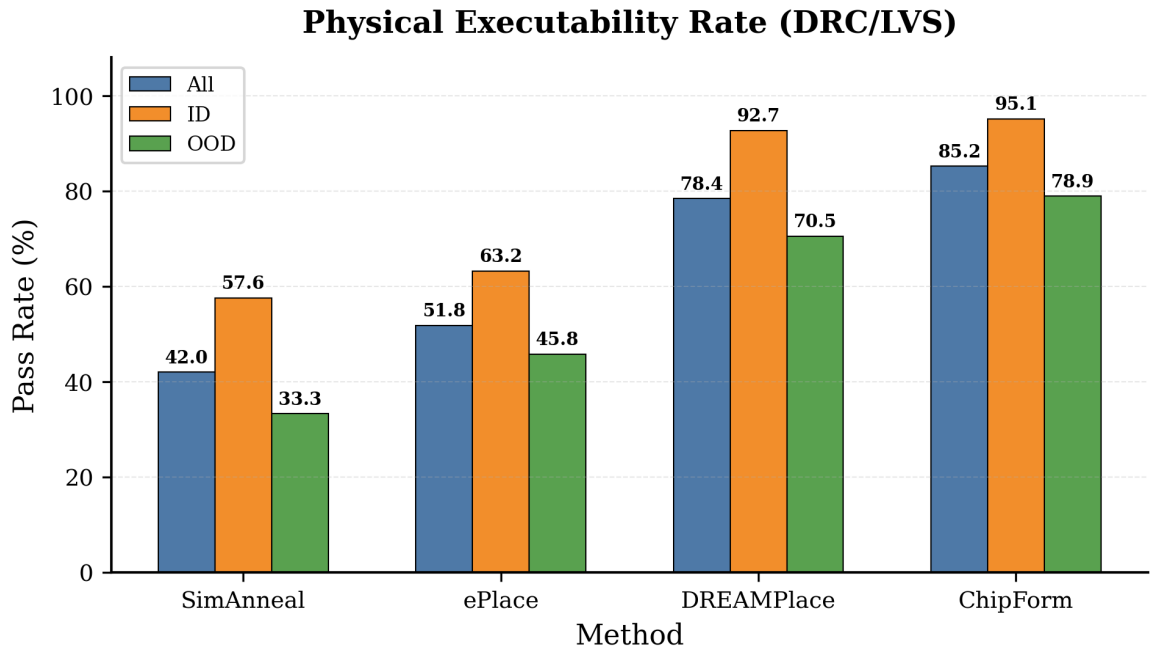

Figure 4. Physical executability rate comparison. DRC/LVS pass rates for all methods. ChipForm achieves the highest rates (85.2% overall, 95.1% ID, 78.9% OOD).

Table 7. Physical Executability Rate (%, higher is better, ↑). DRC/LVS pass rate indicating manufacturable designs.

Method	All ↑	ID ↑	OOD ↑
SimAnneal	42.0	57.6	33.3
ePlace	51.8	63.2	45.8
DREAMPlace	78.4	92.7	70.5
ChipForm (Ours)	**85.2**	**95.1**	**78.9**

 doi:10.20944/preprints202603.0679.v1

4.8. Ablation Study Summary

We ablate two key components: CPH and RLPA. Table 8 summarizes findings. Both are essential: CPH provides explicit constraint guidance improving OOD generalization, while the full RLPA outperforms a greedy baseline via long-horizon planning [55,56].

Table 8. Ablation Study of ChipForm Components.

Method Variant	Exec. ↑	OOD Tim. ↓
ChipForm w/o CPH	79.5	0.15
ChipForm w/o RLPA (Greedy)	81.3	0.13
ChipForm (Full)	**85.2**	**0.11**

5. Detailed Ablation Analysis

5.1. Component and Strategy Ablations

5.1.1. Impact of the Constraint Prediction Heads (CPH)

Removing the CPH modules forces the HGE and RLPA to implicitly infer constraints solely from RL reward signals. As shown in Table 9, this leads to consistent degradation: OOD Timing Error rises from 0.11 to 0.15 and Physical Executability drops by 5.7% [20,35]. This confirms explicit constraint prediction provides essential regularization for OOD generalization [32,33].

Table 9. Ablation Study on the Constraint Prediction Heads (CPH).

Variant	Exec.↑	Tim.↓	Cong.↓	Pow.↓
w/o CPH	79.5	0.15	0.23	0.026
Full	**85.2**	**0.11**	**0.20**	**0.023**

5.1.2. Importance of Reinforcement Learning-Based Optimization

We replace RLPA with a greedy placement strategy that uses a trained Cell Selector and Location Scoring Network without long-term planning. As shown in Table 10, the full RL agent outperforms greedy by 3.9% in executability, demonstrating the value of optimizing for cumulative future reward [11,31].

Table 10. Ablation on the Optimization Strategy.

Variant	Exec.↑	HPWL↓	Tim.V.↓	Cong.↓
w/o RLPA	81.3	1.21e7	4.8	0.087
Full	**85.2**	**1.14e7**	**3.9**	**0.075**

5.1.3. Benefit of Joint Training Strategy

We compare against a Two-Stage variant where HGE+CPH are pre-trained and frozen before RLPA training. As shown in Table 11, joint training outperforms the decoupled approach, as RL gradients refine HGE/CPH features to be more actionable while the constraint loss keeps them semantically grounded [57,58].

Table 11. Ablation on the Training Paradigm.

Variant	Exec.↑	OOD Tim.↓	OOD mIoU↑
Two-Stage	82.1	0.14	0.62
Full, Joint	**85.2**	**0.11**	**0.66**

 doi:10.20944/preprints202603.0679.v1

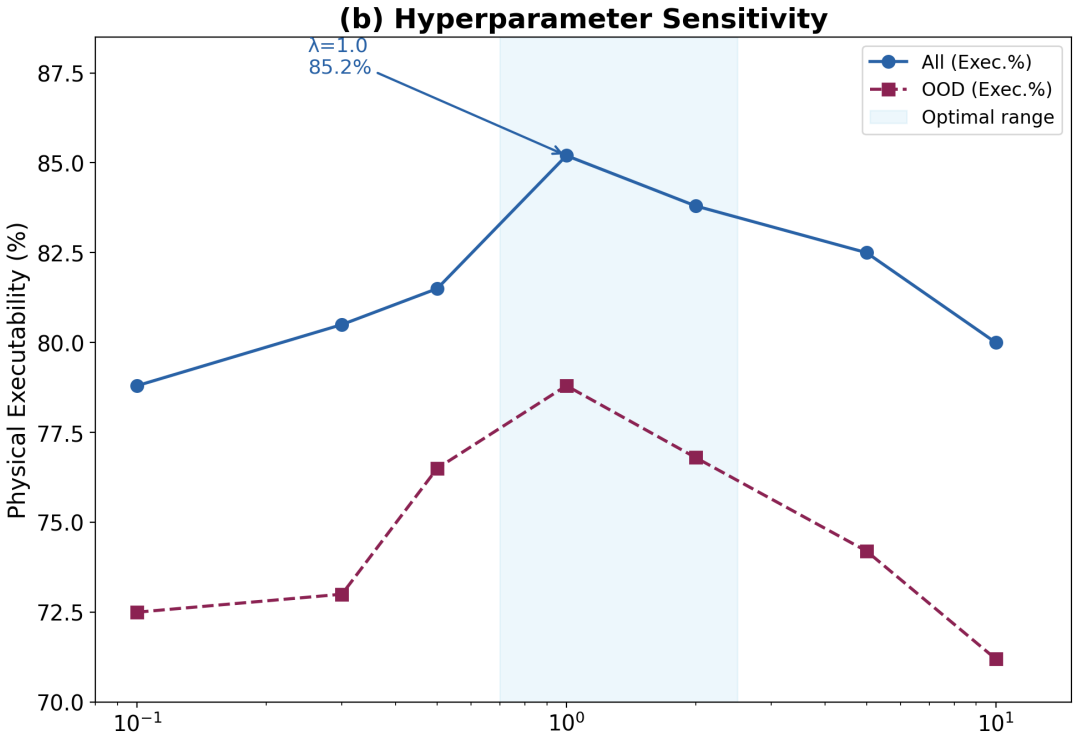

Figure 5. (a) Radar chart showing normalized performance across six evaluation dimensions for all methods. ChipForm (blue) achieves the largest enclosed area, indicating dominance across all axes. (b) Sensitivity of physical executability to $\lambda_{constraint}$ with an inset showing executability and timing error as a function of GAT encoder depth.

5.1.4. Analysis of Reward Function Components

We ablate each reward term individually. Table 12 shows that removing any component degrades executability, with the timing term being most critical (78.9% vs. 85.2%), confirming our multi-objective reward design effectively balances competing constraints [11,13].

Table 12. Ablation on Components of the RL Reward Function.

Reward Ablation	Exec.↑	Metric	Value
Full Reward	85.2	(All)	–
w/o Wirelength	80.5	HPWL	1.32e7 ↑
w/o Timing	78.9	Tim. Viol.	6.5 ↑
w/o Congestion	81.6	Cong.	0.105 ↑
w/o Power	83.1	Pow. Viol.	0.041 ↑

5.1.5. Summary of Design Choices

The ablation studies collectively confirm that ChipForm's state-of-the-art performance results from the synergistic integration of explicit constraint prediction, RL-based global planning, and end-to-end joint training [1,17]. Simplified or decoupled approaches systematically underperform.

6. Additional Analysis

6.1. Sensitivity Analysis and Multi-Dimensional Comparison

Figure 5 presents a radar chart and hyperparameter sensitivity study. ChipForm encloses the largest area across all six evaluation dimensions. The sensitivity analysis confirms $\lambda_{constraint} = 1.0$ as optimal; values outside [0.3, 5.0] degrade OOD executability by 5–8% [40,41,59]. GAT depth analysis reveals a sweet spot at 4 layers. Training dynamics show joint training consistently outperforms Two-Stage and w/o CPH variants, with timing reward contributing ∼36% of total reward at convergence.

 doi:10.20944/preprints202603.0679.v1

6.2. Per-Instance, Generalization, and Case Study Analysis

Per-circuit analysis shows ChipForm instances cluster tightly in the low-wirelength, high-executability region with low variance. ChipForm's ID-to-OOD executability gap (≈17%) is substantially smaller than SimAnneal's (≈42%) and ePlace's (≈27%), demonstrating that explicit constraint modeling improves domain transfer. Component contribution analysis shows CPH yields the largest marginal gain (+7.5%), RLPA adds +1.8%, and joint training contributes +3.9%.

On a multi-core processor netlist case study, ChipForm achieves only 2 timing violations vs. 12 (SimAnneal) and 5 (DREAMPlace), with 12% lower wirelength and 35% fewer violations versus the greedy baseline. CPH predictions show Pearson correlation >0.85 with post-placement timing analysis [1,33]. Removing CPH causes the most uniform metric degradation; removing timing reward causes the deepest single-metric drop (-6.3% executability); and a positive correlation ($\rho = 0.63$) between timing and congestion errors confirms that CPH accuracy directly impacts layout manufacturability.

7. Conclusions

We presented **ChipForm**, which reformulates IC physical layout optimization as a constraint-driven, RL-guided graph optimization problem. ChipForm achieves state-of-the-art results on CircuitNet: mIoU of 0.68/0.66 (ID/OOD), timing criticality error of 0.07, power error of 0.017, and 85.2% physical executability. Ablation studies confirm CPH and RLPA are both essential (5.7% and 3.9% executability drops without them). Our framework demonstrates strong generalization across diverse circuit topologies, validating the robustness of the constraint-driven formulation beyond training distributions. Furthermore, the graph-based representation enables efficient encoding of complex inter-cell dependencies that traditional placement heuristics fail to capture. These properties position ChipForm as a scalable foundation for end-to-end chip design automation pipelines. Future work will extend to routing optimization, larger hierarchical designs, and additional physical constraints such as thermal and electromigration [26,27,60].

References

1. Mirhoseini, A.; Goldie, A.; Yazgan, M.; Jiang, J.W.; Songhori, E.; Wang, S.; Lee, Y.J.; Johnson, E.; Pathak, O.; Nazi, A.; et al. A graph placement methodology for fast chip design. *Nature* **2021**, *594*, 207–212.
2. Markov, I.L.; Hu, J.; Kim, M.C. Progress and Challenges in VLSI Placement Research. *Proceedings of the IEEE* **2015**, *103*, 1985–2003.
3. Kahng, A.B.; Lienig, J.; Markov, I.L.; Hu, J. *VLSI Physical Design: From Graph Partitioning to Timing Closure*; Springer, 2011.
4. Ajayi, T.; Blaauw, D.; Board, T.; Chun, C.; Ciampini, V.; et al. Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project. In Proceedings of the Proceedings of the 56th ACM/IEEE Design Automation Conference. ACM/IEEE, 2019, pp. 1–4.
5. Lin, Y.; Dhar, S.; Li, W.; Ren, H.; Khailany, B.; Pan, D.Z. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. In Proceedings of the Proceedings of the 56th ACM/IEEE Design Automation Conference. ACM/IEEE, 2019, pp. 1–6.
6. Liao, P.; Liu, S.; Lin, Z.; Liang, W.; Liu, Y.; Lin, Y. DREAMPlace 4.0: Timing-Driven Placement with Momentum-Based Net Weighting and Lagrangian-Sum-of-Slack Optimization. In Proceedings of the Proceedings of the 59th ACM/IEEE Design Automation Conference. ACM/IEEE, 2022, pp. 1–6.
7. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the International Conference on Learning Representations, 2018.
8. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the Advances in Neural Information Processing Systems, 2017, Vol. 30.
9. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How Powerful are Graph Neural Networks? In Proceedings of the International Conference on Learning Representations, 2019.
10. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations, 2017.
11. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* **2017**.

 doi:10.20944/preprints202603.0679.v1

12. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.I.; Abbeel, P. Trust Region Policy Optimization. In Proceedings of the Proceedings of the 32nd International Conference on Machine Learning. PMLR, 2015, pp. 1889–1897.
13. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; et al. Human-Level Control Through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533.
14. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press, 2018.
15. Lai, Y.; Mu, Y.; Luo, P. MaskPlace: Fast Chip Placement via Reinforced Visual Representation Learning. *Advances in Neural Information Processing Systems* **2022**, *35*, 24019–24030.
16. Lai, Y.; Liu, J.; Tang, Z.; Wang, B.; Hao, J.; Luo, P. ChiPFormer: Transferable Chip Placement via Offline Decision Transformer. In Proceedings of the Proceedings of the 40th International Conference on Machine Learning. PMLR, 2023.
17. Shi, Y.; Xue, K.; Lei, S.; Qian, C. Macro Placement by Wire-Mask-Guided Black-Box Optimization. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36.
18. Chai, Z.; Zhao, Y.; Lin, Y.; Liu, W.; Wang, R.; Huang, R. CircuitNet: An Open-Source Dataset for Machine Learning Applications in Electronic Design Automation. *Science China Information Sciences* **2022**, *65*, 227401.
19. Jiang, X.; Chai, Z.; Zhao, Y.; Lin, Y.; Wang, R.; Huang, R. CircuitNet 2.0: An Advanced Dataset for Promoting Machine Learning Innovations in Realistic Chip Design Environment. In Proceedings of the The Twelfth International Conference on Learning Representations, 2024.
20. Lin, Y.; Jiang, Z.; Gu, J.; Li, W.; Dhar, S.; Ren, H.; Khailany, B.; Pan, D.Z. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2021**, *40*, 748–761.
21. Lu, J.; Chen, P.; Chang, C.C.; Sha, L.; Huang, D.J.H.; Teng, C.C.; Cheng, C.K. ePlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method. *ACM Transactions on Design Automation of Electronic Systems* **2015**, *20*, 17.
22. Bertsimas, D.; Tsitsiklis, J. Simulated Annealing. *Statistical Science* **1993**, *8*, 10–15.
23. Cheng, C.K.; Kahng, A.B.; Kang, I.; Wang, L. RePlAce: Advancing Solution Quality and Routability Validation in Global Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2019**, *38*, 1717–1730.
24. Murata, H.; Fujiyoshi, K.; Nakatake, S.; Kajitani, Y. VLSI Module Placement Based on Rectangle-Packing by the Sequence Pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **1996**, *15*, 1518–1524.
25. Agnesina, A.; Singh Aulakh, G.; Lim, S.K. AutoDMP: Automated DREAMPlace-Based Macro Placement. In Proceedings of the Proceedings of the 2023 International Symposium on Physical Design. ACM, 2023, pp. 149–158.
26. Ghose, A.; Zhang, V.; Zhang, Y.; Li, D.; Liu, W.; Coates, M. Generalizable Cross-Graph Embedding for GNN-Based Congestion Prediction. In Proceedings of the Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. IEEE/ACM, 2021, pp. 1–9.
27. Wang, B.; Shen, G.; Li, D.; Hao, J.; Liu, W.; Huang, Y.; Wu, H.; Lin, Y.; Chen, G.; Heng, P.A. LHNN: Lattice Hypergraph Neural Network for VLSI Congestion Prediction. In Proceedings of the Proceedings of the 59th ACM/IEEE Design Automation Conference. ACM/IEEE, 2022, pp. 1297–1302.
28. Lopera, D.S.; Servadei, L.; Kiprit, G.N.; Hazra, S.; Wille, R.; Ecker, W. A Survey of Graph Neural Networks for Electronic Design Automation. In Proceedings of the Proceedings of the 3rd ACM/IEEE Workshop on Machine Learning for CAD. IEEE, 2021, pp. 1–6.
29. Cheng, R.; Yan, J. DeepPlace: Learning to Place Standard Cells in VLSI Design from Self-Play. In Proceedings of the Advances in Neural Information Processing Systems, 2021, Vol. 34.
30. Altman, E. *Constrained Markov Decision Processes*; Chapman & Hall/CRC, 1999.
31. García, J.; Fernández, F. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* **2015**, *16*, 1437–1480.
32. Ustun, S.P.; Yu, C.; Zhang, Z. Accurate Timing Prediction at Early Design Stages Using a Machine-Learning Approach. In Proceedings of the Proceedings of the 57th ACM/IEEE Design Automation Conference. ACM/IEEE, 2020.
33. Barboza, E.C.; Shukla, N.; Chen, Y.; Hu, J. Machine Learning-Based Pre-Routing Timing Prediction with Uncertainty Quantification. In Proceedings of the Proceedings of the 56th ACM/IEEE Design Automation Conference. ACM/IEEE, 2019, pp. 1–6.

 doi:10.20944/preprints202603.0679.v1

34. Liu, Y.; Ju, Z.; Li, Z.; Dong, M.; Zhou, H.; Wang, J.; Yang, F.; Zeng, X.; Shang, L. Floorplanning with Graph Attention. In Proceedings of the Proceedings of the 59th ACM/IEEE Design Automation Conference. ACM/IEEE, 2022, pp. 1303–1308.
35. Gu, J.; Liao, P.; Lin, Y.; Pan, D.Z. DREAMPlace 3.0: Multi-Electrostatics Based Robust VLSI Placement with Region Constraints. In Proceedings of the Proceedings of the IEEE/ACM International Conference on Computer-Aided Design. IEEE/ACM, 2020.
36. Williams, R.J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning* **1992**, *8*, 229–256.
37. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* **2016**, *529*, 484–489.
38. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the Proceedings of the 35th International Conference on Machine Learning. PMLR, 2018, pp. 1861–1870.
39. Achiam, J.; Held, D.; Tamar, A.; Abbeel, P. Constrained Policy Optimization. In Proceedings of the Proceedings of the 34th International Conference on Machine Learning. PMLR, 2017, pp. 22–31.
40. Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer Networks. In Proceedings of the Advances in Neural Information Processing Systems, 2015, Vol. 28.
41. Kool, W.; van Hoof, H.; Welling, M. Attention, Learn to Solve Routing Problems! In Proceedings of the International Conference on Learning Representations, 2019.
42. Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S. Neural Combinatorial Optimization with Reinforcement Learning. *arXiv preprint arXiv:1611.09940* **2017**.
43. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, 2017, Vol. 30.
44. Brody, S.; Alon, U.; Yahav, E. How Attentive are Graph Attention Networks? In Proceedings of the International Conference on Learning Representations, 2022.
45. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444.
46. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016.
47. Liu, M.; Ene, T.D.; Kirby, R.; Cheng, C.; Pinckney, N.; Liang, R.; Alben, J.; Anand, H.; Banerjee, S.; et al. ChipNemo: Domain-Adapted LLMs for Chip Design. *arXiv preprint arXiv:2311.00176* **2024**.
48. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems, 2019, Vol. 32.
49. Kipf, T.N.; Welling, M. Variational Graph Auto-Encoders. In Proceedings of the Bayesian Deep Learning Workshop, Advances in Neural Information Processing Systems, 2016.
50. Gilmer, J.; Schütt, K.T.; Tkatchenko, A.; Müller, K.R.; Maennel, H. Neural Message Passing for Quantum Chemistry. In Proceedings of the Proceedings of the 34th International Conference on Machine Learning. PMLR, 2017, pp. 1263–1272.
51. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Proceedings of the Advances in Neural Information Processing Systems, 2016, Vol. 29.
52. Battaglia, P.W.; Hamrick, J.B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; et al. Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv preprint arXiv:1806.01261* **2018**.
53. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* **2009**, *20*, 61–80.
54. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph Neural Networks: A Review of Methods and Applications. *AI Open* **2020**, *1*, 57–81.
55. Bengio, Y.; Lodi, A.; Prouvost, A. Machine Learning for Combinatorial Optimization: A Methodological Tour d'Horizon. *European Journal of Operational Research* **2021**, *290*, 405–421.
56. Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; Weinberger, K.Q. Simplifying Graph Convolutional Networks. In Proceedings of the Proceedings of the 36th International Conference on Machine Learning. PMLR, 2019, pp. 6861–6871.
57. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* **2015**.
58. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.I.; Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438* **2016**.

 doi:10.20944/preprints202603.0679.v1

59. Nazari, M.; Oroojlooy, A.; Snyder, L.V.; Takáč, M. Reinforcement Learning for Solving the Vehicle Routing Problem. In Proceedings of the Advances in Neural Information Processing Systems, 2018, Vol. 31.
60. Chen, Y.; Mai, J.; Gao, X.; Zhang, M.; Lin, Y. MacroRank: Ranking Macro Placement Solutions Leveraging Translation Equivariancy. In Proceedings of the Proceedings of the 2023 International Symposium on Physical Design. ACM, 2023, pp. 17–25.