iPCL: Pre-training for Chip Layout

(Invited Paper)

Xingquan Li[1,✉], Weiguo Li[1], Xinhua Lai[2], Junfeng Liu[1], Rui Wang[3],
[1]Pengcheng Laboratory, [2]University of Chinese Academy of Sciences, [3]Shenzhen University
✉fzulxq@gmail.com

***Abstract*—As chip complexity continues to grow, traditional rule-based EDA tools face increasing challenges in optimizing power, performance, and area (PPA). There is a pressing need for a foundation model in chip layout to leverage large-scale historical design data for unified and intelligent physical design. We propose iPCL (pre-training for chip layout), a comprehensive framework that integrates placement and routing generation, metric evaluation and optimization. iPCL consists of layout symbolization, multimodal pre-training, solution generation and selection, and post-processing stages, forming a scalable and automated design pipeline. iPCL reduces design iteration time, supports multi-layout generation, and automatically selects optimal solutions through lightweight ECO refinement. Two versions are developed: iPCL-R, which generates routing layouts with performance comparable to commercial tools while reducing design time by 55.7%; and iPCL-M, which delivers 336× faster and more accurate metric evaluation than open-source EDA, achieving about 5% better optimization results than commercial tools.**

***Index Terms*—Chip layout, pre-training, foundation model, symbol representation, placement and routing**

I. Introduction

Physical design is a key stage in electronic design automation (EDA), transforming synthesized netlists into manufacturable layouts. It includes placement, routing, timing closure, and design-rule checking (DRC), which jointly determine chip performance, power, and area (PPA). Traditional flows rely on heuristic algorithms, analytical solvers, and static timing analysis (STA) to optimize design metrics under complex constraints [1]. Placement typically uses wirelength estimations such as HPWL and FLUTE-based Steiner models [2], [3], while routing depends on maze and rip-up-and-reroute heuristics. Despite decades of advances, these methods remain computationally expensive and technology-dependent, limited by rule complexity and the exponential growth of design search space [4], [5].

Recent progress in machine learning (ML) and small-model AI has introduced data-driven approaches for layout evaluation and optimization. These models aim to approximate time-consuming design analyses such as wirelength, delay, or congestion estimation without invoking full signoff engines [6], [7]. Traditional ML regressors and neural surrogates can predict timing metrics using engineered RC features or pre-routing topologies [8]. More advanced graph-based methods leverage circuit connectivity via graph neural networks (GNNs) or graph transformers to estimate delay and congestion [9]–[12]. Deep learning frameworks further enhance inference speed for design-space exploration [13], [14]. However, these small models are often task-specific, data-dependent, and passive: they can efficiently evaluate metrics but lack the ability to synthesize or optimize layouts across multiple objectives simultaneously [15], [16].

Inspired by breakthroughs in large language models (LLMs) and foundation models, the EDA community has begun to explore large-model paradigms to unify and automate design workflows. Transformer-based architectures [17], [18] have been adapted for chip design, leading to systems such as MOSS for RTL-level timing optimization [19], ChatEDA for natural-language-driven tool orchestration [20], and multi-agent systems for analog and digital co-design [21], [22]. These early attempts demonstrate strong reasoning and generalization ability, yet they mostly rely on natural-language or code-level tokens, which fail to capture the geometric and hierarchical nature of layouts [23]–[25]. Large circuit models (LCMs) [26] emphasizes multimodal circuit representation learning, integrating diverse data sources such as functional specifications, RTL designs, netlists, and physical layouts for a more holistic design understanding. Building a *domain-grounded foundation model* that can represent and generate layout data at the physical level—analogous to how LLMs model text—remains a fundamental challenge and opportunity for next-generation EDA intelligence [27]–[29].

Motivated by this gap, we propose **iPCL**, a representation-centric pre-training framework for chip layout generation. As illustrated in Fig. 1, iPCL employs a structured *layout symbol system* that encodes cells and nets as symbolic primitives with geometric semantics. These symbols are tokenized for self-supervised pre-training, enabling the model to learn layout generation and reconstruction from large-scale corpora. After pre-training, iPCL can be fine-tuned for placement, routing, and ECO optimization, serving as a scalable foundation model that enhances layout automation and design efficiency.

The contributions can be summarized as follows:

- We propose a pre-training–based foundation model for chip layout (iPCL) that integrates symbolic design, self-supervised pre-training, and layout generation with post-processing, enabling efficient, scalable, and EDA-compatible layout generation, evaluation, and optimization.
- We propose a layout symbol vocabulary system that represents multiple layout modalities with minimal information loss and enables their serialization, providing the foundation for designing diverse pre-training tasks to learn layout semantics.
- We build a pre-training model for routing (iPCL-R) that generates fabrication-ready routing results with a lightweight ECO, achieving comparable wirelength, power, and timing to commercial tools while reducing runtime by 55.7%.
- We propose a pre-trained model for metric evaluation and optimization (iPCL-M) that is 336× faster and more accurate than iEDA, achieving approximately 5% improvement

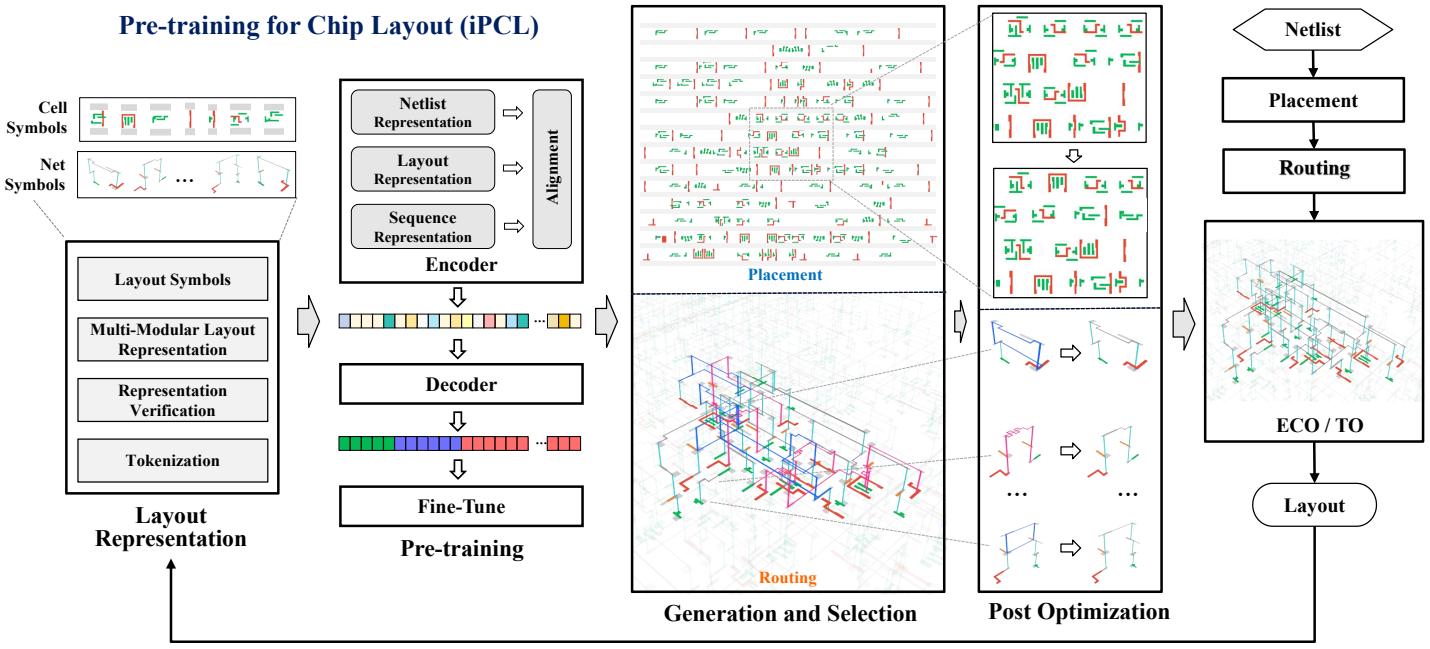


Fig. 1. The framework of iPCL.

over commercial tools in design optimization.

II. Layout Decomposition and Overview

This section introduces chip design decomposition and our pre-training based generative framework for chip layout.

A. Chip Layout Decomposition

Regarding the results of routing, we can decompose the layout into several levels of components, as shown in Fig. 5 of AiEDA [16]. **Netlist and Layout**: Back-end chip design consists of two core components: the netlist defining logical connections between instances, and the layout containing physical manufacturing patterns. **Layer and Row**: Modern multi-layer chips organize metal layers into structured rows or panels for efficient placement and routing. Each metal layer can be conceptually treated as a two-dimensional image for processing. **Net**: As the fundamental unit in physical design, a net connects multiple pins across instances, with the two-pin net representing the simplest and most common connection type. **Path**: For timing and power analysis, critical paths are extracted from the netlist, each comprising a sequence of interconnected two-pin nets that form complete signal propagation routes. **Wire**: The most abundant routing element, confined to a single metal layer. Wire characteristics (width, layer) directly determine RC parasitics, significantly impacting overall power and performance. **Via**: Essential vertical connectors that enable signal transitions between layers. Each via includes a cut layer opening with associated top/bottom metal shapes, with different via types offering distinct electrical and physical properties. **Pin**: Cell interface components containing multiple rectangular shapes across layers. Internally, pins follow defined signal propagation logic; externally, they connect to other cells through wires and vias via nets. **Patch**: Optimization elements introduced as rectangular shapes on specific layers to address timing, IR drop, or capacitance constraints, improve signal integrity, and minimize additional routing resource consumption.

B. Overview of Pre-training for Chip Layout (iPCL)

Traditional chip layout design starts from a netlist and proceeds through placement and routing to generate the layout, which is further refined via engineering change order (ECO) based on post-layout evaluations. This process relies heavily on EDA tools and manual expertise, requiring multiple time-consuming iterations to achieve design closure.

To address these limitations, we propose pre-training for chip layout (iPCL), a generative paradigm that enables layout design without manual intervention or EDA tool dependence. iPCL offers three key benefits: (1) it reduces design iteration time from hours to minutes, (2) enables concurrent generation of multiple layouts with a high probability of outperforming training samples, and (3) achieves fabrication-ready designs by selecting the best layout followed by a lightweight ECO process. As shown in Fig. 1, iPCL comprises five stages: (1) **Layout Representation:** decomposition and symbolic encoding of historical layouts; (2) **Pre-training:** multimodal representation learning using an encoder–decoder or decoder-only model; (3) **Generation and Selection:** concurrent layout generation and optimal solution selection; (4) **Post Optimization:** optional refinement of generated layouts; (5) **Flow Integration:** embedding the refined layout into the conventional flow for final signoff.

In the layout representation stage, chip layouts are decomposed into atomic symbols (cells, wires, vias, and pins), represented across multiple modalities (graph, image, sequence), verified for information fidelity, and tokenized for learning. In the pre-training stage, the tokenized data are used to learn layout semantics through generative modeling, followed by fine-tuning for specific layout design, evaluation, and optimization tasks.

III. Vocabulary and Layout Representation

We define the basic layout structure and a tailored symbol vocabulary, extracting key instances and routing elements—wires, locations, and polygons—for tokenized representation.

TABLE I
SYMBOLS USED FOR CHIP LAYOUT REPRESENTATION.

Component	Symbol	Meaning
Cell	NAND2_X1	Cell of a 2-input NAND gate with drive strength X1
Pin	I_A	Input pin labeled "A" within a cell instance
Wire	M3_H100	Horizontal wire with 100nm on metal 3
Via	V12_M3	1 × 2 via connecting metal layers 3 and 4

A. Symbol for Layout Components

Inspired by natural language, where letters form words and words encode compositional meaning, we develop a symbolic representation for chip layouts to capture their fundamental elements. We construct a layout symbol system that encodes geometric entities and connectivity as unified symbols, representing wires, locations, and topology. This symbolic abstraction preserves spatial semantics more effectively than raw coordinates and provides a foundation for serialization and sequence-based generative modeling.

1) Symbolic Space for Chip Layout: In a chip layout, the primary elements—cell, pin, wire, and via (Table I)—are represented as parameterized symbols. Each layout can be decomposed into these fundamental, reusable symbols, capturing both structural semantics and geometric variability. For instance, cell symbols encode dimensions, orientation, or threshold voltage; wire symbols include metal layer, length, direction, and width; via symbols connect layers with defined rules; and pin symbols indicate signal or power connections. By encoding parameters within each symbol, complex layouts are transformed into a discrete yet expressive form, enabling unified multimodal representation learning and generative modeling.

2) Wire and Via Symbolization: In iPCL, routing wires and vias are treated as physical *connection symbols* linking pins. Each connection is abstracted as a sequence of directed movements in 3D space, defined by *direction* and *displacement*, transforming continuous routing data into discrete symbolic patterns for sequence modeling and pretraining. A connection path, or *wire pattern*, consists of movement commands with six directions: Right (`R`), Left (`L`), Up (`U`), Down (`D`) in the x–y plane, and Top (`T`) and Bottom (`B`) along the z-axis. For example, "`R2000`" is a 2000-unit rightward segment, and "`T2`" is a two-layer via. Displacement values are further tokenized into numeric symbols (0–9), so "`R2000`" becomes "`R2 0 0 0`". This yields a compact, expressive vocabulary for both planar and vertical routing. Formally, let $\mathcal{D} = \{\mathrm{R}, \mathrm{L}, \mathrm{U}, \mathrm{D}, \mathrm{T}, \mathrm{B}\}$ denote movement directions and $\{0, \ldots, 9\}$ numeric symbols for displacement, forming the symbol space for wire and via representation.

3) Location Symbolization: To provide a consistent spatial reference, we define a fixed origin $o = (0, 0, 0)$ and describe vertices, pins, or terminals relative to this origin or other points using direction and numeric symbols. In iPCL, the direction symbols used for wires and vias (`R`, `L`, `U`, `D`, `T`, `B`) are reused for location representation, allowing each point to be encoded as a sequence of relative displacements. For instance, $(x, y, z) = (400, 300, 2)$ can be represented as [`U300`, `R400`, `T2`]. Direct absolute coordinates lead to sparse and less generalizable data. To address this, iPCL adopts *relative displacement encoding*, representing each location as the positional difference between two reference points (e.g., driver and load pins). This approach reduces vocabulary size, preserves connectivity semantics, and enables more stable, transferable learning. Formally, for $p_s = (x_s, y_s, z_s)$ and $p_t = (x_t, y_t, z_t)$, the displacement $\Delta p = p_t - p_s$ is symbolized using the same direction–distance notation, unifying position and routing representations in a consistent vocabulary for pre-training and layout generation.

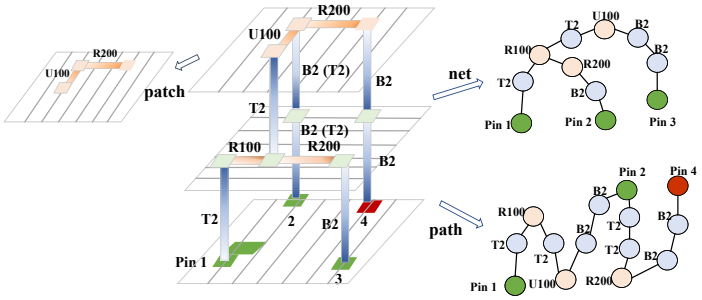


Fig. 2. A chip layout and its corresponding multi-modular structures (net, path, patch) represented in the symbolic space.

4) Topology Symbolization: Beyond geometry, *topological connectivity* among layout components is symbolically represented to preserve circuit structure. In iPCL, each routed net n_i is encoded via depth-first traversal (DFS) of its routing tree, capturing geometry and hierarchy in a unified symbolic form. Wire and via directions are represented by direction–length tokens, while topological relationships use control symbols: `BRANCH` marks fan-out points, `END` indicates branch completion, `PUSH` inserts a sub-path, `POP` exits it, and `BOS`/`EOS` denote sequence boundaries. This tokenization enables full reconstruction of net hierarchy without storing the graph explicitly, reducing memory overhead while preserving structural semantics.

B. Multi-modular Layout Representation

Fig. 2 illustrates the interconnection among four layout instances (pins) in a chip design, where pin 1 serves as the source. According to the symbol space defined earlier, the connection from pin 1 to pin 2 is encoded as `[T2, R100, T2, U100, B2, B2]`. Here, `T2` and `B2` denote vertical displacements of two units upward and downward; `R100` indicates a horizontal displacement of 100 units to the right; and `U100` represents an upward movement of 100 units. This symbolic space offers a unified and flexible way to represent layout data. In Fig. 2, three representative data modules describe the design: **Path (sequence):** A timing path connects two two-pin nets (pin 1–pin 2 and pin 2–pin 4). This path is represented as a sequence using the wire and via vocabulary: `[DRIVER, T2, R100, T2, U100, B2, B2, LOAD, T2, T2, R200,B2, B2, LOAD]`. **Net (graph):** Pins 1, 2, and 3 belong to a single net. Its topology is captured through the same symbolic vocabulary as `[DRIVER, T2, R100, BRANCH, R200, B2, LOAD, BRANCH, T2, U100, B2, B2, LOAD]`. **Region (patch):** Several wire segments in metal layer 3 are shown in Fig. 2. Their spatial distribution is abstracted as a two-dimensional patch, e.g., `[U100, R200]`.

To illustrate the advantages of symbolic representation, we compare coordinate-based and symbol-based net encodings (Fig. 3). Coordinate-based representation stores each vertex as (x, y, z), which is intuitive but introduces numerical redundancy

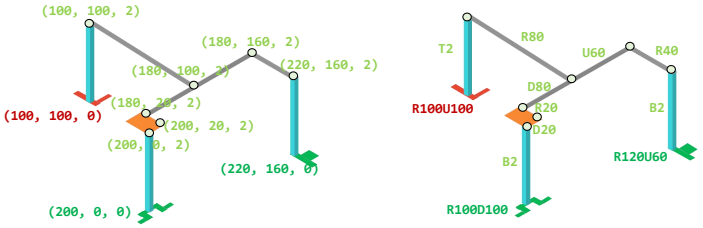


Fig. 3. Illustration of a net. (a) Net representation by coordinates. (b) Net representation by symbols

```
{
  "Driver": "(100,100,0)",
  "Loads": ["(200,0,0)","(220,160,0)"],
  "Routing": ["(100,100,0)","(100,100,2)","(180,100,2)",
     "[BRANCH]","(180,20,2)","(200,20,2)","(200,0,2)","(200,0,0)","[END]",
     "[BRANCH]","(180,160,2)","(220,160,2)","(220,160,0)","[END]" ]
}
```

(a) Coordinate-based

```
{
  "Source Tokens": ["<BOS>","<DRIVER>","R100U100",
     "<LOAD>","R100D100","<LOAD>","R120U60","<SRC_END>"],
  "Target Tokens": ["<EOS>","T2","R80",
     "<PUSH>","D80","R20","D20","B2","<POP>",
     "<PUSH>","U60","R40","B2","<POP>","<EOS>"]
}
```

(b) Symbol-based

Fig. 4. (a) Coordinate-based net representation and generation. (b) Symbol-based net representation and generation.

and ambiguous semantics, complicating tokenization and model learning. As shown in Fig. 4(a), driver and load positions can be serialized into a compact symbolic sequence that preserves relative spatial relationships. In contrast, symbol-based representation encodes geometric and topological elements using a compact vocabulary (e.g., "R100", "T2", "[BRANCH]"). This reduces sequence length, ensures discrete tokens, and maintains precise design semantics, producing a deterministic, reversible sequence that captures both geometric continuity and hierarchical topology (Fig. 4(b)). The result is a more interpretable, compact, and generalizable layout representation for data-driven modeling.

Our preliminary study shows that human language tokenizers (e.g., *T5-Gemma-2B*) perform poorly on serialized coordinates, with only 19 active tokens out of 250K (Fig. 5(a)). By contrast, the domain-specific symbolic vocabulary uses far fewer tokens with higher utilization, forming clear directional clusters ($\{R, L\}$, $\{U, D\}$, $\{T, B\}$, Fig. 5(b)) and effectively capturing the spatial semantics of routing layouts for compact, generalizable, and interpretable sequence modeling.

C. Layout Representation Fidelity Verification

Although symbolic layout representations offer compactness and generalization, they may introduce an abstraction gap. To ensure fidelity for model training, we verify them from three perspectives: statistical coverage, spatial reconstruction, and design consistency. **(1) Statistical Coverage.** We analyze geometric and topological attributes (e.g., wire length, via count, layer utilization, fan-out) to confirm that symbolic counterparts preserve dominant trends and variations, demonstrating broad representational coverage across layouts. **(2) Spatial Reconstruction.** Layouts are reconstructed from symbolic sequences into images for pixel-level comparison. Metrics like SSIM and correlation show high spatial fidelity. For example, the reconstructed cell density map of design `s713` (Fig. 6) preserves key spatial features with a correlation of 0.993. **(3) Design Consistency.** Symbolized layouts are re-evaluated in the standard chip flow for timing, power, and DRC. PPA metrics remain highly aligned with

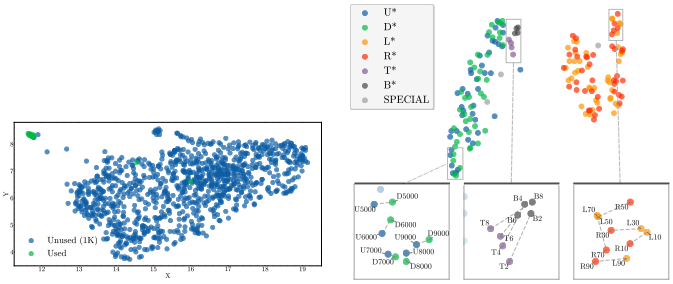


Fig. 5. (a) Coordinate embeddings, with 1,000 symbols randomly sampled due to many unused tokens. (b) Symbol embeddings, where dashed lines show spacial semantic relationships (e.g., 'T4' and 'B4' represent symmetric displacement).

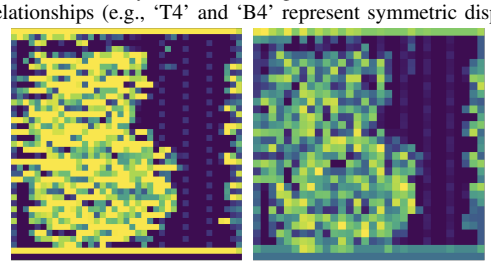


Fig. 6. Patch-level fidelity comparison for the `s713` design.

TABLE II
NET-LEVEL FIDELITY COMPARISON FOR THE `s713` DESIGN.

Metric	Original	Reconstructed	Fidelity Ratio
WNS (ns)	-0.972	-0.989	0.983
TNS (ns)	-3.887	-3.987	0.975
Violating Paths	7	7	1.000
Total Power (W)	0.0621	0.0622	0.998

coordinate-based layouts (Table II), including timing-violating paths and power measures, confirming that critical physical and electrical properties are preserved.

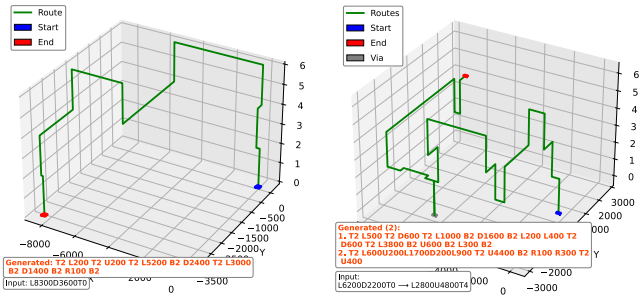
D. Tokenization

The symbolic representation provides a unified space for routing geometries using direction–length symbols, but its effectiveness—compactness, interpretability, and learnability—depends on the *tokenization strategy*. Proper tokenization discretizes continuous routing data into atomic tokens, revealing intrinsic patterns, while improper segmentation can blur geometric intent or introduce redundancy. Decimal tokenization yields balanced frequency distribution and smaller symbol set than byte pair encoding (BPE) or byte-level BPE (BBPE). Unlike frequency-driven merges, the decimal word level (WDL) scheme aligns with the routing grid hierarchy, letting models capture multi-scale regularities in layout geometry. Tokenization bridges symbolic abstraction and spatial reasoning, encoding geometric knowledge for data-driven models. In iPCL, it is a *core principle* for high-fidelity, interpretable layout learning.

IV. Pre-training for Chip Layout

A. Pre-training Tasks

We adopt a GPT-style autoregressive Transformer for fully generative layout modeling, leveraging the symbolic representation and unified tokenization. In iPCL, pretraining tasks teach the model geometric regularities of routing and structural semantics of connectivity.



(a) Two pins net (b) Three pins net

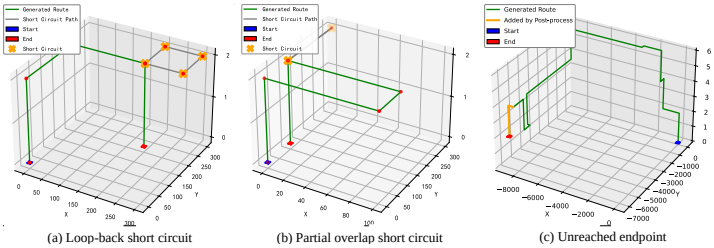
Fig. 7. Routing net generation results in design s5378.

Pretraining is organized into two classes: *Decoder-only* and *Encoder–Decoder*. Decoder-only tasks focus on autoregressive sequence modeling, predicting the next token to capture routing topology, geometric continuity, and design intent, with routing net generation as the primary task. Encoder–Decoder tasks enable context recovery and spatial understanding, learning bidirectional representations for reasoning about both local and global layout structures.

Routing Net Generation: Each routed net is represented as a symbolic sequence of direction–length and control tokens (Section III). The model is trained autoregressively to predict the next token: $p(\mathcal{S}) = \prod_{t=1}^{T} p(s_t|s_{<t})$, where $\mathcal{S} = \{s_1, \ldots, s_T\}$ is the symbolized net sequence. This next-token prediction captures routing syntax and geometry, enabling layout completion, refinement, and net expansion (Fig. 7). **Routing Net Mask and Recovery:** Randomly mask tokens in a net sequence and train the model to recover them, learning topological dependencies and wire continuity. **Layout Mask and Recovery:** Mask layout patches in image space, and predict missing regions from context, enabling cross-modal pretraining and hierarchical spatial understanding. **Layout Denoising:** Inject controlled noise (coordinate jitter, symbol perturbation, or distortion) and train the model to reconstruct clean layouts, improving robustness to imperfect data. **Netlist Mask and Recovery:** Mask nodes or edges in the connectivity graph and train the model to infer missing elements, capturing higher-level logical relationships beyond geometry. **Feature Fitting:** Predict numerical layout or circuit attributes (e.g., wirelength, delay, power, congestion) from symbolic or image representations, aligning learned features with physical EDA metrics.

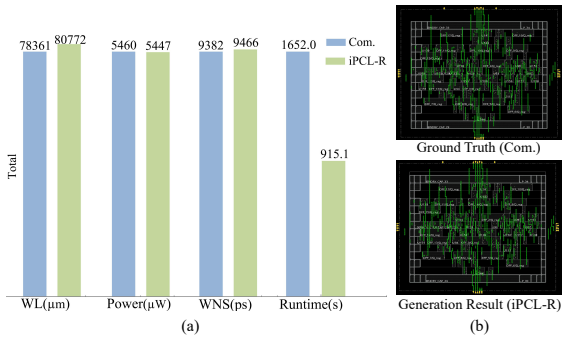
B. Fine-tuning

While pretraining captures general geometric and structural priors from large-scale layouts, downstream chip design tasks require task-specific reasoning. We perform supervised fine-tuning using additional features and labels for objectives like placement refinement, routing optimization, and physical metric prediction. During fine-tuning, the model conditions on symbolic and numerical features (e.g., cell attributes, net criticality, timing slack, congestion maps) to predict outputs such as wirelength, timing, power, or optimized layouts. This aligns the pretrained representation with physical metrics, enabling accurate reasoning and optimization. Compared to training from scratch, fine-tuning accelerates convergence and improves fidelity, allowing iPCL to adapt to diverse applications including layout generation, DRC-aware routing, and PPA-driven optimization.



(a) Loop-back short circuit (b) Partial overlap short circuit (c) Unreached endpoint

Fig. 8. Post-processing cases. (a) and (b) show removed wires (gray) during short-circuit removal; (c) shows an added wire (yellow) to complete the connection.



(a) (b)

Fig. 9. Comparison between the results of Commercial tool and the results generated by iPCL-R.

C. Solution Generation and Post-processing

Once pretrained and fine-tuned, iPCL can generate layout or routing solutions under specific design objectives. For PPA-oriented generation, techniques such as beam search, temperature scaling, and diverse sampling allow exploration of multiple candidates. Parallel solution generation enables multiple designs in a single pass, increasing the chance of high-quality solutions and reducing runtime from hours to minutes. The best candidate is selected for verification and optimization.

If generated layouts violate physical or logical constraints, a post-processing stage ensures correctness using traditional EDA algorithms to fix shorts, unreachable endpoints, or DRC violations (Fig. 8). This hybrid approach combines AI generative power with deterministic EDA, producing efficient, high-quality, and physically realizable layouts.

V. Experimental Results

The experimental environment for the following tasks is as follows. The hardware configuration: CPU (Intel Xeon Platinum 8380 CPU with 160 cores), Memory (512 GB RAM), GPU (NVIDIA A100 with 40 GB VRAM); the software configuration: Operation System (Ubuntu 20.04.6), PyTorch (2.0.1), CUDA (12.0), scikit-learn (1.2.2), pandas (1.5.3), and matplotlib (3.7.1). Data generation was performed by AiEDA [16].

A. Pre-training for Chip Layout Routing (iPCL-R)

iPCL is a pre-training model for chip layout synthesis, primarily used for generating placement and routing solution. This section introduces some results of Pre-training for Chip Layout Routing (iPCL-R). Based on the ECO support and evaluation interfaces of commercial tools, we validated the quality of iPCL-R's generated routing results. As shown in Fig. 9(a), compared to commercial tools (Com.), our iPCL-R achieves almost the same results on wirelength (WL.), power, timing (WNS). Additionally, our runtime is significantly reduced by 55.7%, with inference

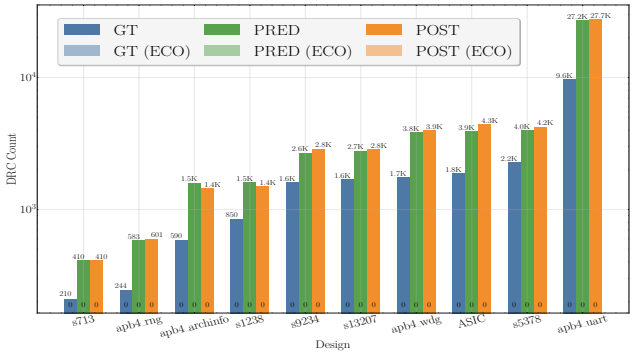


Fig. 10. The DRC distribution for each design, after being fixed through ECO, can achieve DRC clean.

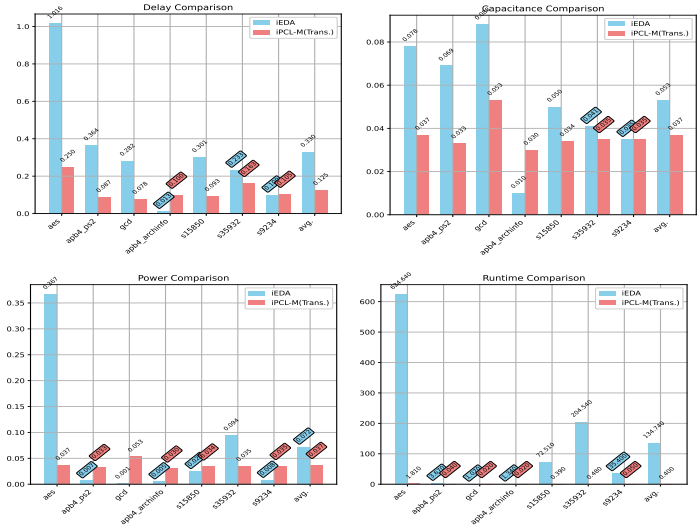


Fig. 11. Comparison of iEDA and iPCL-M on delay, power, capacitance and runtime.

time accounting for only 10.3% of that in traditional commercial workflows. Overall, the results of iPCL-R indicate a higher cost in terms of design resource usage, which may stem from resource usage preferences due to differences in training data distribution. However, in terms of the final PPA metrics for chip design, there is no significant difference, and iPCL-R achieves a notable improvement in runtime. Fig. 9(b) illustrates the layout images of metal layer 2 of design `S713` generated from commercial tool and iPCL-R. It is difficult to notice the difference between these two images, which indicates that iPCL-R is capable of learning the solution of the sample layout.

To further demonstrate the effectiveness of iPCL-R, we evaluated the number of DRC violations. As shown in Fig. 10, DRC violations were counted for the ground truth (GT), iPCL-R predictions (PRED), and optimized results after post-processing (POST), including counts following ECO. The number of violations for GT, PRED, and POST remain within the same order of magnitude and scale proportionally with design size. All designs achieved DRC-clean status after ECO, indicating that the current generation paradigm can produce high-quality routing.

B. Pre-training for Chip Layout Metric Evaluation and Optimization (iPCL-M)

In addition to generating placement and routing solutions, iPCL can also be used to evaluate and optimize design metrics in the physical design process. We build a metric evaluation and optimization version of iPCL (iPCL-M).

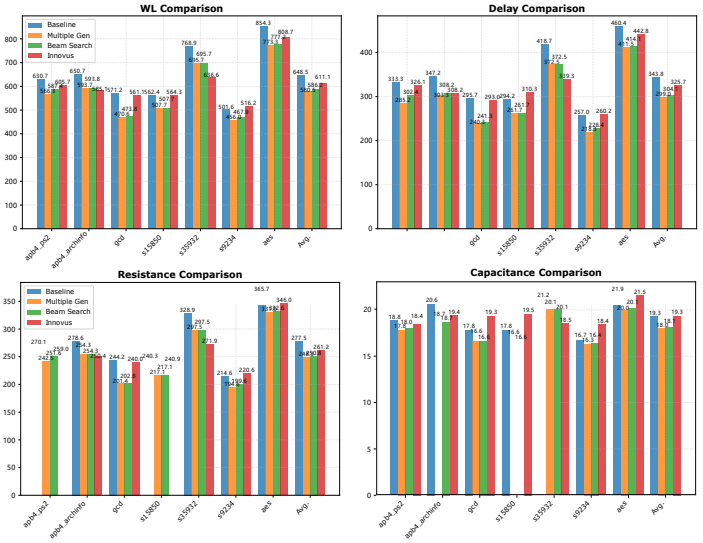


Fig. 12. Comparison of iPCL-M (baseline, multiple generation, beam search) and Innovus on wirelength, delay, resistance, capacitance.

Fig. 11 presents the RMSE between the predicted values (delay, capacitance, and power) from iEDA and the iPCL-M model, compared to the ground-truth values from Innovus. Since iEDA derives metrics from routed data rather than predicting them directly, these metrics are excluded from the statistics. The results indicate that iPCL-M generally outperforms iEDA across most metrics. In terms of inference time, iEDA requires 624.6s and 204.54s for large designs such as `aes` and `s35932`, respectively, whereas iPCL-M only needs 1.81s and 0.48s, achieving an average 336× speedup (iEDA: 134.74s, iPCL-M: 0.4s). These findings demonstrate that iPCL-M provides more accurate predictions for delay, capacitance, and power, while substantially improving computational efficiency.

Furthermore, we evaluate the optimization capability of our iPCL-M model. As shown in Fig. 12, we compare multiple generation (iPCL-M), beam search (iPCL-M), the baseline model (iPCL-M), and the commercial tool Innovus. The baseline represents results without optimization. From the column "Avg.", multiple generation improves wirelength, delay, resistance, and capacitance by 10.49%, 13.03%, 10.51%, and 6.46%, respectively, compared to the baseline. Beam search achieves improvements of 9.61%, 11.55%, 9.62%, and 6.18%, respectively. Innovus yields 5.77%, 5.26%, 5.85%, and -0.22%, respectively. Overall, Innovus underperforms multiple generation by approximately 5% across all four metrics, likely because our optimization does not consider design rule checks (DRC).

VI. Conclusion

In this work, we present a pre-training framework for chip layout (iPCL) that bridges AI foundation models and physical design automation. By learning from large-scale historical data, iPCL unifies layout generation, metric evaluation, and optimization within a single model. Experiments demonstrate its strong generalization and efficiency: iPCL-R achieves commercial-level routing quality with less design time, while iPCL-M delivers much faster and more accurate metric evaluation than open-source EDA tool, achieving better optimization results than commercial EDA tool. These results highlight iPCL's potential to advance intelligent, scalable, and high-quality chip layout design.

REFERENCES

[1] T.-W. Huang and M. D. F. Wong, "OpenTimer: A high-performance timing analysis tool," in *Proc. of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 895–902.

[2] B. Obermeier and F. M. Johannes, "Quadratic placement using an improved timing model," in *Proc. of the 41st annual Design Automation Conference (DAC)*, 2004, pp. 705–710.

[3] J. Huang, X.-L. Hong, C.-K. Cheng, and E. S. Kuh, "An efficient timing-driven global routing algorithm," in *Proc. of the 30th international Design Automation Conference (DAC)*, 1993, pp. 596–600.

[4] X. Li, Z. Huang, S. Tao *et al.*, "iEDA: An Open-source Infrastructure of EDA," in *Proc. of IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024, pp. 77–82.

[5] T. Ajayi, V. A. Chhabria, M. Fogaça *et al.*, "INVITED: Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project," in *Proc. of IEEE/ACM Design Automation Conference (DAC)*, 2019, pp. 1–4.

[6] V. A. Chhabria, W. Jiang, A. B. Kahng, and S. S. Sapatnekar, "A machine learning approach to improving timing consistency between global route and detailed route," *ACM Transactions on Design Automation of Electronic Systems*, vol. 29, no. 1, pp. 1–25, 2023.

[7] Y. Li, R. Liu, Z. Zeng *et al.*, "AiDRC: Accelerating Detailed Routing by AI-Driven Design Rule Violation Prediction and Checking," *ACM Transactions on Design Automation Electronic Systems*, 2025.

[8] X. He, Z. Fu, Y. Wang, C. Liu, and Y. Guo, "Accurate Timing Prediction at Placement Stage with Look-Ahead RC Network," in *Proc. of the 59th ACM/IEEE Design Automation Conference (DAC)*, 2022, p. 1213–1218.

[9] Z. Xie, R. Liang, X. Xu, J. Hu, Y. Duan, and Y. Chen, "Net2: A graph attention network method customized for pre-placement net length estimation," in *Proc. of the 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021, pp. 671–677.

[10] L. Liu, B. Lu, Y. Li, L. Shang, and F. Yang, "GTN-Path: Efficient Path Timing Prediction through Waveform Propagation with Graph Transformer," in *Proc. of 2025 62nd ACM/IEEE Design Automation Conference (DAC)*, 2025, pp. 1–7.

[11] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin, "A timing engine inspired graph neural network model for pre-routing slack prediction," in *Proc. of the 59th ACM/IEEE Design Automation Conference (DAC)*, 2022, pp. 1207–1212.

[12] H. Liu, Z. Zeng, S. Tao, Y. Li, B. Xie, W. Gao, and X. Li, "AiTPO: KAN-UNet Heterogeneous Network for Timing Prediction and Optimization at Global Routing," *ACM Transactions on Design Automation Electronic Systems*, 2025.

[13] J. Ahn, K. Chang, K.-M. Choi, T. Kim, and H. Park, "DTOC-P: Deep-Learning-Driven Timing Optimization Using Commercial EDA Tool With Practicality Enhancement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 8, pp. 2493–2506, 2024.

[14] X. Lai, M. Liu, X. Li, Y. Qiu, S. Chen, X. Li, and J. Xu, "iPO: Constant Liar Parameter Optimization for Placement with Representation and Transfer Learning," *ACM Transactions on Design Automation Electronic Systems*, 2025.

[15] V. A. Chhabria, W. Jiang, A. B. Kahng *et al.*, "OpenROAD and CircuitOps: Infrastructure for ML EDA Research and Education," in *Proc. of IEEE VLSI Test Symposium (VTS)*, 2024, pp. 1–4.

[16] Y. Qiu, Z. Huang, S. Tao, H. Zhang, W. Li, X. Lai, R. Wang, W. Wang, and X. Li, "AiEDA: An Open-source AI-Aided Design Library for Design-to-Vector," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[19] M. Wang, B. Sun, J. Mu *et al.*, "MOSS: Multi-Modal Representation Learning on Sequential Circuits," in *Proc. of 2025 62nd ACM/IEEE Design Automation Conference (DAC)*, 2025, pp. 1–7.

[20] Z. He, H. Wu, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu, "ChatEDA: A large language model powered autonomous agent for EDA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 10, pp. 3184–3197, 2024.

[21] B. Liu, H. Zhang, X. Gao, Z. Kong, X. Tang, Y. Lin, R. Wang, and R. Huang, "LayoutCopilot: An LLM-Powered Multiagent Collaborative Framework for Interactive Analog Layout Design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 8, pp. 3126–3139, Aug. 2025.

[22] H. Wu, H. Zheng, Z. He, and B. Yu, "Divergent Thoughts toward One Goal: LLM-based Multi-Agent Collaboration System for Electronic Design Automation," *arXiv preprint arXiv:2502.10857*, 2025.

[23] K. Chang, K. Wang, N. Yang *et al.*, "Data is all you need: Finetuning LLMs for Chip Design via an Automated design-data augmentation framework," in *Proc. of the 61st ACM/IEEE Design Automation Conference (DAC)*, Nov. 2024, pp. 1–6.

[24] M. Liu, T.-D. Ene, R. Kirby, C. Cheng *et al.*, "ChipNeMo: Domain-Adapted LLMs for Chip Design," *arXiv preprint arXiv:2311.00176*, 2024.

[25] J. Pan, I. Jacobson, Z. Zhao, T.-C. Chen, G. Zhou, C.-C. Chang, V. Rashingkar, and Y. Chen, "CROP: Circuit Retrieval and Optimization with Parameter Guidance using LLMs," *arXiv preprint arXiv:2507.02128*, 2025.

[26] L. Chen, Y. Chen, Z. Chu *et al.*, "Large circuit models: opportunities and challenges," *Science China Information Sciences*, vol. 67, no. 10, p. 200402, 2024.

[27] W. Fang, J. Wang, Y. Lu, S. Liu, and Z. Xie, "GenEDA: Unleashing Generative Reasoning on Netlist via Multimodal Encoder-Decoder Aligned Foundation Model," *arXiv preprint arXiv:2504.09485*, 2025.

[28] W. Fang, J. Wang, Y. Lu, S. Liu, Y. Wu, Y. Ma, and Z. Xie, "A Survey of Circuit Foundation Model: Foundation AI Models for VLSI Circuit Design and EDA," *arXiv preprint arXiv:2504.03711*, 2025.

[29] S. Li, X.-H. Li, F. Yin, and L.-L. Huang, "Region-level layout generation fornbsp;multi-level pre-trained model based visual information extraction," in *Pattern Recognition: 27th International Conference, ICPR 2024, Kolkata, India, December 1–5, 2024, Proceedings, Part XIX*, 2024, p. 234–249.