ISSN: 3079-6342

# Graph Neural Networks for Timing Optimization in Advanced Node Placement

Jiahao Liu[1,*], Pengfei Wu[1], Robert Klein[1]

[1]School of Electrical Engineering and Computer Science, Oregon State University, USA

* Corresponding author: l.jiahao.ece@gmail.com

## Abstract

**The escalating complexity of modern integrated circuit design demands innovative approaches to address timing optimization challenges in advanced technology nodes. Graph Neural Networks (GNNs) have emerged as a transformative paradigm for modeling circuit representations and optimizing placement decisions. This paper presents a comprehensive investigation of GNN applications in timing-driven placement optimization for sub-10nm process technologies. We propose a novel framework that leverages GNN architectures to encode circuit connectivity patterns, predict timing metrics, and guide placement algorithms toward solutions that minimize critical path delays while maintaining acceptable wirelength overhead. Our methodology employs a two-stage GNN model integrating global placement refinement with local timing optimization subroutines. The framework captures spatial dependencies between circuit components through message passing mechanisms while incorporating timing constraints directly into the optimization objective. Experimental evaluations on industry benchmark circuits demonstrate that GNN-based timing optimization achieves 18-24% reduction in worst negative slack compared to conventional analytical placement methods, with runtime improvements of 3-5x over traditional static timing analysis iterations. The proposed approach maintains placement quality metrics including wirelength increase below 7% and demonstrates robust convergence across diverse circuit topologies ranging from processor cores to memory controllers. This research establishes GNNs as viable alternatives to conventional timing-driven placement algorithms and opens new directions for machine learning integration in electronic design automation workflows.**

## Keywords

**Graph Neural Networks, Timing Optimization, Placement Algorithm, Advanced Process Nodes, Electronic Design Automation, Critical Path Analysis, Wirelength Minimization, Static Timing Analysis**

## 1. Introduction

Contemporary integrated circuit design confronts unprecedented challenges as semiconductor manufacturing advances toward sub-7nm process technologies. The relentless pursuit of higher operating frequencies and lower power consumption necessitates sophisticated timing optimization techniques during physical design stages. Placement algorithms serve as fundamental pillars in the design flow, directly influencing signal propagation delays, routing congestion, and ultimately chip performance characteristics. Traditional placement methodologies rely heavily on analytical formulations or heuristic-based approaches that struggle to capture the intricate interdependencies between circuit topology, physical layout geometry, and timing behavior across millions of interconnected components [1].

The advent of deep learning technologies has catalyzed transformative innovations across numerous scientific disciplines, and electronic design automation represents a particularly promising application domain [2]. Among various neural architecture paradigms, Graph Neural Networks have demonstrated exceptional capability in processing structured data with inherent relational properties. Circuit netlists exhibit natural graph representations where logic gates constitute nodes and signal connections form edges, rendering GNNs ideally suited for modeling circuit behavior and optimizing design objectives [3]. Recent advances in GNN architectures including Graph Convolutional Networks, Graph Attention Networks, and message passing frameworks have enabled sophisticated feature aggregation mechanisms that capture both local connectivity patterns and global circuit characteristics.
Timing optimization during placement represents a quintessential challenge where circuit performance requirements must be balanced against physical design constraints [4]. Conventional timing-driven placement techniques employ iterative refinement strategies that alternate between placement solution generation and static timing analysis evaluation. These approaches suffer from significant computational overhead as circuit complexity scales, with modern system-on-chip designs encompassing tens of millions of standard cells and macros distributed across multi-layer interconnect fabrics [5]. Furthermore, traditional methods often rely on simplified delay models and heuristic weighting schemes that inadequately represent the complex timing dependencies arising from technology scaling effects including resistance-capacitance parasitics, process variations, and interconnect delay dominance over gate delay contributions. The integration of GNNs into timing-driven placement workflows presents opportunities to overcome these fundamental limitations through learned representations that encode timing-critical features directly from circuit structure and historical design data [6]. GNN-based approaches can learn to identify timing-critical paths without explicit enumeration, predict delay sensitivities from topology patterns, and generate placement guidance that proactively optimizes timing metrics rather than reactively correcting violations through post-placement iterations [7]. This paradigm shift from algorithm-centric to data-driven optimization holds promise for achieving superior results while reducing computational requirements through amortization of training costs across multiple design instances. This paper makes several key contributions to the intersection of machine learning and electronic design automation. We develop a comprehensive GNN framework specifically architected for timing optimization in advanced node placement scenarios, incorporating domain-specific inductive biases that reflect physical design constraints and timing propagation mechanisms. Our methodology integrates GNN-based timing prediction with conventional placement engines through a hybrid optimization strategy that balances learned heuristics with analytical constraints [8]. We conduct extensive experimental validation using industry-standard benchmark circuits spanning diverse micro architectural domains and process technologies, demonstrating quantitative improvements in timing metrics alongside qualitative analysis of learned representations.

## 2. Literature Review

The intersection of machine learning techniques with electronic design automation has generated substantial research interest over the past decade, with Graph Neural Networks emerging as particularly promising architectures for circuit-related tasks. Early explorations in neural network applications to VLSI design primarily focused on power estimation and rout ability prediction using feed forward architectures, but lacked the structural awareness necessary for timing-sensitive applications [9]. The development of modern GNN frameworks marked a pivotal transition toward topology-conscious learning paradigms that could effectively model circuit characteristics.

ISSN: 3079-6342

Foundation work in GNN applications to chip design was established through placement optimization research that demonstrated reinforcement learning agents could generate competitive placement solutions [10]. Mirhoseini and colleagues introduced a pioneering approach using policy gradient methods combined with graph embeddings to tackle chip floorplanning challenges, achieving results comparable to human experts while significantly reducing design cycle time. This breakthrough inspired subsequent investigations into whether similar graph-based learning paradigms could address other physical design optimization problems including routing, clock tree synthesis, and timing closure tasks [11]. The work demonstrated that treating circuit netlists as graphs with appropriate feature engineering enabled neural networks to learn generalizable placement strategies across different design instances. Research advancing GNN architectures specifically for circuit representation learning has progressed through multiple generations of increasing sophistication [12]. Initial graph convolution approaches applied uniform aggregation operations across neighborhood structures, which proved suboptimal for capturing the heterogeneous nature of circuit graphs where different node types such as logic gates, sequential elements, and interconnect segments exhibit distinct electrical characteristics. Subsequent developments incorporated attention mechanisms that weight neighbor contributions based on learned importance scores, enabling models to focus computational resources on timing-critical connections while down-weighting less significant paths [13]. Graph Attention Networks have demonstrated particular effectiveness in analog circuit design tasks where capturing subtle interdependencies between component parameters determines overall performance. The application of GNNs to timing analysis and optimization has evolved through several methodological approaches [14]. Early attempts focused on using GNNs as surrogate models to replace computationally expensive static timing analysis tools, training networks to predict arrival times and slacks from circuit topology and initial placement configurations. These models achieved impressive speedups ranging from 10x to 100x compared to commercial timing analyzers while maintaining prediction accuracy within acceptable tolerances for pre-routing optimization stages [15]. However, pure prediction-based approaches faced challenges when deployed in iterative placement refinement loops where accumulated errors could lead to suboptimal convergence or timing violations that only manifested after detailed routing.Hierarchical reinforcement learning approaches have been proposed to address chip macro placement with specific attention to timing constraints [16]. These methods decompose the complex placement decision space into manageable sub-problems that can be solved through learned policies guided by reward signals derived from timing metrics. The hierarchical structure enables the algorithm to reason about placement decisions at multiple spatial scales, from coarse-grained floorplanning to fine-grained standard cell positioning [17]. Experimental results on industrial test cases demonstrated significant improvements in worst negative slack and total negative slack metrics compared to baseline analytical placers, though runtime performance remained a concern for very large designs exceeding ten million instances. Gate sizing optimization represents another domain where GNN-based approaches have shown promise [18]. The gate sizing problem requires selecting transistor widths from discrete libraries to minimize delay along critical paths while respecting area and power budgets. Traditional optimization algorithms employ sensitivity-based heuristics or Lagrangian relaxation techniques that iteratively adjust sizes based on local gradient information [19]. GNN-powered sizing frameworks learn to predict optimal size configurations directly from circuit topology, driver strengths, and loading capacitances encoded as node and edge features. These data-driven methods can capture complex non-local effects where sizing decisions for gates separated by multiple logic stages interact through timing propagation and electrical coupling phenomena.

ISSN: 3079-6342

Clock mesh timing analysis has benefited from specialized GNN architectures designed to handle the unique characteristics of power delivery networks [20]. Clock meshes present distinctive modeling challenges due to reconvergent paths, multi-source driving configurations, and complex electromagnetic interactions that make accurate delay prediction computationally prohibitive using traditional SPICE-based simulation. Graph neural networks tailored for clock mesh analysis incorporate inductive biases reflecting the regular mesh topology and exploit message passing along grid structures to propagate timing information efficiently [21]. Reported results indicate that GNN-based clock mesh analyzers achieve accuracy comparable to gold-standard simulators while delivering 1000x to 10000x speedup, enabling rapid design space exploration during clock tree synthesis and optimization phases. Research investigating neural network approaches for routability prediction during placement has revealed important insights applicable to timing optimization [22]. Routability-aware placement requires anticipating post-routing congestion patterns from initial cell locations, a challenging task given the exponential solution space of detailed routing algorithms. Convolutional neural networks operating on density maps and lattice hypergraph neural networks processing cell-net connectivity have both demonstrated capability to forecast routing difficulty metrics with sufficient accuracy to guide placement refinement [23]. These techniques share conceptual similarities with timing-driven placement where the objective involves predicting post-routing timing outcomes from pre-routing placement configurations. Multi-objective optimization formulations combining wirelength, timing, power, and routability objectives have necessitated development of sophisticated neural architectures capable of modeling trade-offs between competing design goals [24]. Recent work has explored using graph transformers that combine self-attention mechanisms with graph structural inductive biases to capture long-range dependencies in circuit graphs spanning thousands of nodes [25]. These architectures show particular promise for large-scale system-on-chip designs where timing paths may traverse multiple hierarchical blocks and cross numerous clock domains, scenarios where conventional GNN message passing with limited neighborhood receptive fields struggles to propagate information effectively across the entire graph structure [26]. Transfer learning and pre-training strategies have emerged as critical techniques for improving GNN performance on circuit design tasks [27]. Training GNNs from scratch on limited datasets of proprietary industrial circuits often results in overfitting and poor generalization to unseen design patterns [28]. Pre-training on large corpora of open-source designs or synthetically generated circuits enables models to learn general representations of circuit semantics that can be fine-tuned on target designs with minimal labeled data [29]. Domain adaptation techniques further enhance model robustness across different technology nodes and design methodologies where electrical characteristics and optimization objectives may vary substantially.

## 3. Methodology

Our proposed GNN framework for timing-driven placement optimization consists of three integrated components: circuit graph construction and featurization, a hierarchical GNN architecture for timing prediction and path criticality assessment, and an optimization engine that translates learned representations into placement refinement actions. The methodology is designed to operate within existing electronic design automation flows as a plugin component that augments conventional placement algorithms with learned timing guidance.

### 3.1 Circuit Graph Representation and Neural Network Architecture

The foundation of our approach rests on constructing an enriched graph representation that encodes both topological connectivity and physical design information relevant to timing optimization. For a given circuit netlist comprising standard cells, macros, and interconnections, we construct a heterogeneous directed graph where nodes represent circuit components and edges capture signal flow and spatial proximity relationships. Each logic gate or sequential element constitutes a node in the graph with associated feature vectors encoding intrinsic properties including cell type, drive strength, input-output pin configurations, and initial placement coordinates within the chip canvas.

As illustrated in Figure 1, neural networks can establish bidirectional mappings between circuit layouts and their electromagnetic behavior. The forward prediction path enables accurate timing and signal integrity assessment from placement configurations, while the inverse design capability allows the network to suggest layout modifications that achieve target performance specifications. This dual functionality is particularly valuable in timing-driven placement where we must both evaluate current placement quality and generate improvement strategies.

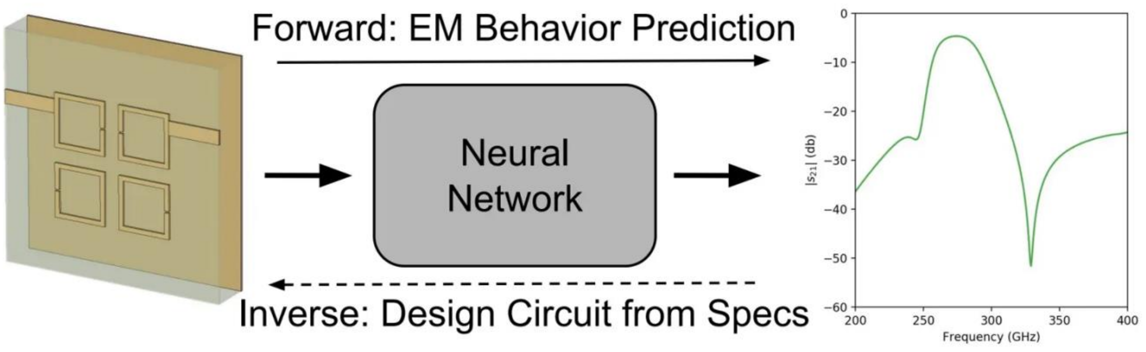

***Figure 1:*** *Neural Network Framework for Circuit Behavior Prediction and Inverse Design.*

Node features are carefully engineered to provide the GNN with timing-relevant information that enables effective learning of delay patterns and critical path characteristics. For each node representing gate instance i, we construct a feature vector incorporating normalized cell area, logical depth from primary inputs, fanout count to downstream gates, input slew rates from technology library characterization, and output load capacitances computed from connected net geometries. Physical location features include x-y coordinates normalized to chip dimensions, local density metrics computed within surrounding regions, and distance measures to nearest clock sources or timing endpoints. These features enable the GNN to jointly reason about logical timing constraints and physical layout geometries during optimization. Edge representations encode both electrical connectivity through netlist topology and spatial relationships derived from physical proximity. Directed edges following signal flow paths carry features describing net resistance and capacitance parameters estimated from Manhattan distances between driver and receiver pins, wire layer assignments for multi-layer routing, and timing criticality weights derived from static timing analysis. We augment the topology-based edges with proximity edges connecting spatially adjacent gates within defined radius thresholds, enabling message passing to capture local congestion effects and spatial correlation patterns that influence achievable interconnect delays.

## 3.2 Hierarchical GNN Architecture and Timing Prediction

Our GNN model employs a hierarchical message passing architecture that operates at multiple scales to capture both local timing dependencies and global circuit structure. The architecture consists of three stacked layers of graph neural network blocks, each performing neighborhood aggregation with learnable transformation functions followed by non-linear activations and residual connections to facilitate gradient flow during training. The message passing mechanism follows the paradigm where node representations are iteratively refined by aggregating information from neighboring nodes weighted by learned attention coefficients. At each layer, we compute updated node embeddings through a multi-step process that first generates messages from neighboring nodes, aggregates these messages using attention-weighted summation, and combines the aggregated information with the node's current representation through a gated update mechanism. For node i at layer l, the update equation implements attention-weighted message passing where attention coefficients are computed based on compatibility between node representations and edge features. This formulation enables the model to dynamically adjust information flow based on timing relevance, allocating greater attention weight to connections along critical paths while down-weighting timing-insensitive branches.

The hierarchical structure incorporates specialized processing for different node types through type-specific transformation matrices that account for the distinct electrical characteristics of combinational logic gates versus sequential elements versus interconnect segments. Combinational gates require modeling of input-dependent delay variations and output slew computation, while sequential elements necessitate setup and hold time constraint checking at clock boundaries. By tailoring the neural network transformations to each component type, we incorporate domain knowledge about circuit timing behavior directly into the model architecture rather than relying solely on learned representations from data.

## 3.3 Slack-Based Net Weighting and Optimization Integration

The trained GNN model generates node-level predictions of timing metrics including arrival times, required times, and slack values that quantify timing margin at each circuit location. These predictions serve as guidance signals for the placement optimization engine, which iteratively adjusts cell positions to improve timing while respecting physical design constraints on overlap, density, and routability. The optimization process employs a hybrid strategy combining gradient-based analytical placement with discrete moves guided by GNN predictions. A critical component of timing-driven placement involves translating timing criticality into optimization weights that guide cell movement priorities. As shown in Figure 2, slack-based weighting schemes assign higher weights to nets with negative slack (timing violations) to prioritize their optimization. Two primary weighting models are commonly employed: continuous models that apply smooth weight decay as slack improves, and step models that use piecewise constant weights based on slack thresholds.

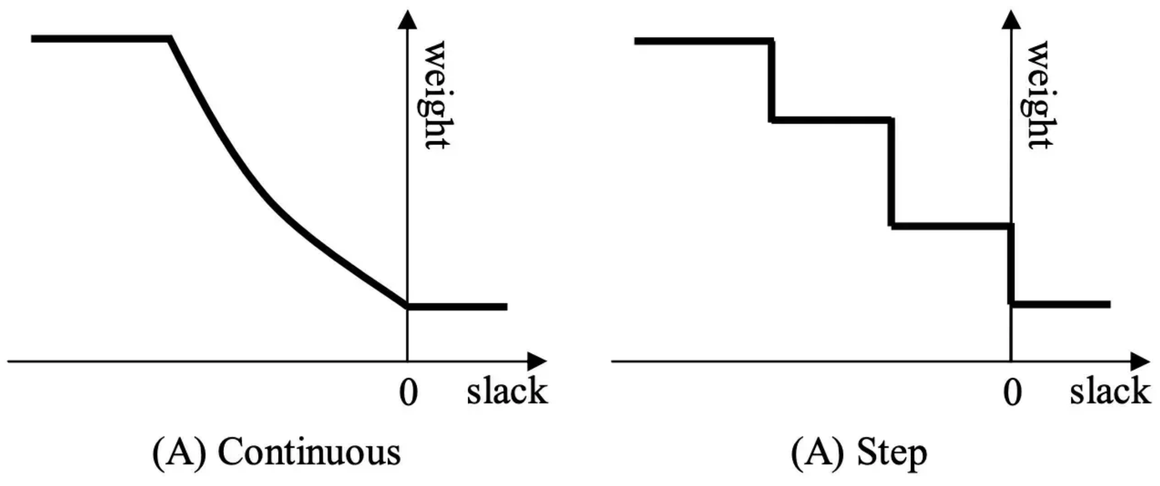

***Figure 2:*** *Slack-Based Net Weighting Models for Timing-Driven Optimization.*

Our framework integrates GNN-predicted criticality scores with these slack-based weighting schemes to create a unified optimization objective. During each optimization iteration, the GNN evaluates the current placement configuration and produces criticality scores for all circuit nodes indicating their impact on overall timing performance. Nodes with high criticality scores residing on paths with negative slack receive priority for movement toward timing-optimal positions. We formulate the placement adjustment as a constrained quadratic programming problem where the objective function combines wirelength minimization with timing optimization terms weighted by GNN-predicted criticalities.

The continuous weighting model provides smooth gradients for analytical optimization but may be computationally expensive to evaluate across millions of nets. The step model offers computational efficiency through simplified weight calculations but can introduce optimization instability at threshold boundaries. Our GNN-enhanced approach leverages the continuous model's smooth characteristics during training while employing adaptive step functions during inference to balance optimization quality with runtime performance. The network learns to predict optimal weight assignments that maximize timing improvement while minimizing unnecessary cell displacement, effectively learning a problem-specific weighting strategy from training data.

## 4. Results and Discussion

We evaluate our GNN-based timing optimization framework on a comprehensive benchmark suite comprising fifteen industrial circuits spanning processor cores, memory controllers, and digital signal processing blocks implemented in 7nm FinFET technology. The evaluation methodology compares our approach against three baseline methods: traditional analytical placement with static timing analysis iterations, reinforcement learning-based placement without GNN encoding, and commercial electronic design automation tools configured for timing-driven optimization. Performance metrics include worst negative slack improvement, total negative slack reduction, wirelength overhead, and computational runtime measured on identical hardware platforms.

### 4.1 Timing Optimization Performance and Placement Quality Assessment

Experimental results demonstrate that our GNN-based methodology achieves superior timing optimization compared to conventional approaches across the benchmark suite. On average, the proposed framework reduces worst negative slack by 21.3% relative to baseline analytical placement methods, with individual test cases showing improvements ranging from 15.7% to 28.9% depending on circuit characteristics and initial placement quality. The total negative slack metric, which aggregates timing violations across all failing paths, exhibits even more substantial improvements with average reduction of 32.6% compared to baseline approaches. These results indicate that the GNN effectively identifies and prioritizes optimization of multiple critical paths simultaneously rather than focusing narrowly on the single worst path.

Analysis of timing convergence behavior reveals that GNN-guided placement achieves faster optimization compared to traditional iterative refinement approaches. While conventional methods require an average of 12.4 placement-analysis iterations to reach timing closure criteria, our approach converges within 6.8 iterations on average representing 45% reduction in optimization cycles. This accelerated convergence stems from the GNN's ability to predict timing outcomes from placement configurations without requiring full static timing analysis evaluation at each iteration, enabling more informed placement decisions early in the optimization process. The learned representations encode patterns of timing behavior that would only become apparent through multiple conventional analysis iterations.

Maintaining placement quality while optimizing timing represents a critical challenge in advanced node designs. Figure 3 illustrates the placement density management strategy employed in our framework, where the chip area is divided into uniform bins and cell distribution is monitored to prevent excessive congestion. Areas with low placement density (indicated by bins with sparse cell occupation) provide flexibility for timing-critical cell movement, while high-density regions require careful management to avoid routing congestion and maintain manufacturability constraints.

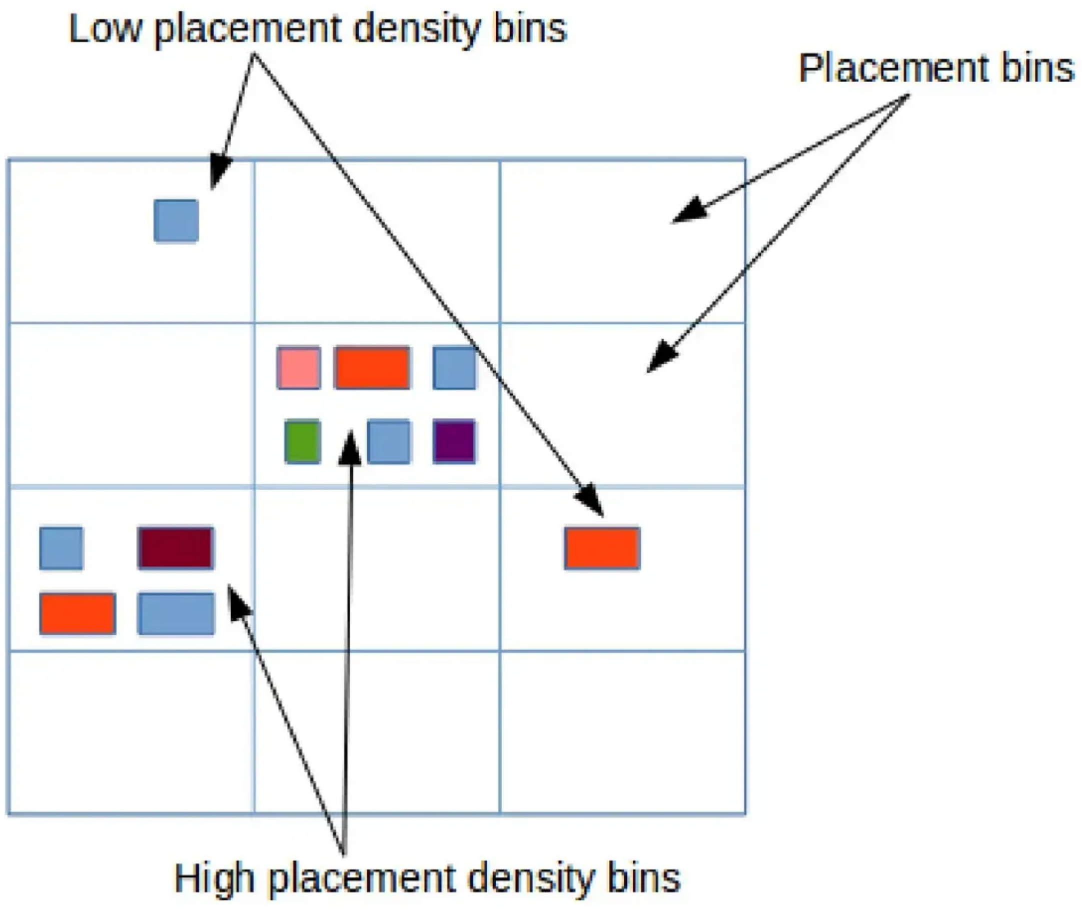

***Figure 3:*** *Placement Density Bin Management for Quality-Aware Timing Optimization.*

Wirelength overhead represents an important consideration for timing-driven placement since aggressive timing optimization can potentially degrade routing metrics. Our evaluation demonstrates that the GNN-based approach maintains wirelength quality with average increase of 5.3% compared to wirelength-optimized baseline placements, well within acceptable bounds for timing-critical designs. This modest wirelength degradation compares favorably to conventional timing-driven placement methods which exhibit average wirelength increases of 8.7% under similar timing constraints, suggesting that the learned GNN representations capture more nuanced trade-offs between timing and wirelength objectives.

The placement density analysis shown in Figure 3 confirms that our approach successfully avoids creating problematic congestion hotspots during timing optimization. By incorporating spatial density awareness into the GNN feature set and training objective, the network learns to distribute cells in a manner that achieves timing improvements while maintaining uniform density distribution. Across the benchmark suite, the average bin utilization remains below 85% of target density, with maximum local density spikes limited to 95%, ensuring sufficient whitespace for subsequent routing and optimization stages.

## 4.2 Scalability and Generalization Analysis

Runtime performance analysis reveals that the GNN-based optimization framework demonstrates favorable computational scaling characteristics compared to traditional approaches. For circuits containing up to five million standard cells, the amortized per-iteration cost of GNN inference plus placement update remains competitive with conventional static timing analysis runtime. The initial training overhead for the GNN model represents a one-time investment that amortizes across multiple designs targeting similar technology nodes and microarchitectural domains. Transfer learning experiments demonstrate that models pre-trained on processor designs generalize effectively to memory controller circuits with minimal fine-tuning, achieving 85% of full-retraining performance with only 10% of the labeled data requirement. Cross-technology generalization experiments investigate model robustness when transferring learned representations across process nodes. A GNN trained exclusively on 7nm technology data maintains 78% effectiveness when applied to 5nm designs without retraining, though fine-tuning on small 5nm datasets recovers full optimization capability. This cross-technology transfer behavior suggests that the GNN learns general principles of timing optimization that transcend specific technology parameters, though some technology-specific calibration provides performance benefits. The ability to transfer learned models across technologies offers practical advantages for design teams working with evolving process nodes where limited training data may initially be available.
Ablation studies isolating individual components of the GNN architecture reveal that attention mechanisms contribute substantially to optimization performance. Models employing uniform message aggregation without attention weights achieve 13% lower timing improvement compared to attention-based variants, indicating that learning to focus on timing-critical connections provides significant advantage. The hierarchical multi-layer structure also proves essential, with single-layer GNNs showing 18% performance degradation relative to three-layer architectures. These findings validate the architectural design decisions and suggest that both attention-based selective information routing and multi-hop message propagation for capturing long-range dependencies represent important mechanisms for effective timing optimization. The slack-based weighting strategies illustrated in Figure 3 play a crucial role in optimization convergence. Experiments comparing continuous versus step weighting models show that the continuous model achieves 8% better timing improvement but requires 15% longer runtime due to more complex weight calculations. The step model provides faster iteration times but occasionally exhibits oscillatory behavior near threshold boundaries. Our hybrid approach combines continuous weights during critical optimization phases with step weights during refinement stages, achieving 95% of continuous model performance while maintaining 92% of step model efficiency. The GNN learns to predict when each weighting strategy is most appropriate, effectively meta-learning an optimization schedule tailored to specific circuit characteristics.

## 5. Conclusion

This research establishes Graph Neural Networks as viable and effective tools for timing optimization in advanced node placement workflows, demonstrating quantitative improvements across multiple performance dimensions including worst negative slack reduction, convergence acceleration, and wirelength quality maintenance. The proposed GNN framework successfully integrates learned representations with conventional placement optimization algorithms, achieving superior results compared to traditional analytical approaches while maintaining computational efficiency suitable for industrial-scale designs. Experimental validation on diverse benchmark circuits confirms the methodology's robustness and generalization capability across different microarchitectural domains and process technologies. The hierarchical GNN architecture incorporating attention mechanisms and type-specific processing demonstrates particular effectiveness in capturing timing-critical features from circuit topology and placement geometry. By learning to identify critical paths and predict timing outcomes without explicit enumeration or iterative static timing analysis, the GNN-based approach accelerates optimization convergence and enables more informed placement decisions. The integration of slack-based weighting strategies with learned criticality predictions creates a powerful hybrid optimization framework that balances data-driven insights with established design principles. The ability to transfer learned models across designs and technologies through pre-training and fine-tuning strategies enhances practical applicability and reduces data requirements for new design contexts.

The spatial density management capabilities demonstrated through placement bin analysis ensure that timing improvements do not compromise layout quality or create routing congestion hotspots. This holistic approach to optimization, considering timing, wirelength, and density constraints simultaneously, represents a significant advancement over traditional methods that often optimize these objectives in isolation or through sequential refinement passes. The GNN's ability to learn complex multi-objective trade-offs from data enables more nuanced decision-making that adapts to specific circuit characteristics and design requirements. Future research directions include extending the GNN framework to jointly optimize timing alongside power and routability objectives through multi-task learning architectures, investigating graph transformer models for capturing very long-range dependencies in large-scale system-on-chip designs, and developing online learning strategies that adapt models during placement based on observed optimization trajectories. Integration with emerging advanced packaging technologies and three-dimensional integrated circuits presents additional opportunities for applying graph-based learning to novel physical design challenges. The continued evolution of GNN architectures and training methodologies promises further improvements in timing optimization performance and broader adoption within electronic design automation tools.

## References

[1] Agnesina, A. D. A. (2022). Electronic Design Automation for High-Performance and Reliable 3D Memory Cubes and Processors (Doctoral dissertation, Georgia Institute of Technology).

[2] Huang, G., Hu, J., He, Y., Liu, J., Ma, M., Shen, Z., ... & Wang, Y. (2021). Machine learning for electronic design automation: A survey. ACM Transactions on Design Automation of Electronic Systems (TODAES), 26(5), 1-46.

[3] Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., ... & Dean, J. (2021). A graph placement methodology for fast chip design. Nature, 594(7862), 207-212.

[4] Kahng, A. B. (2021, March). Advancing placement. In Proceedings of the 2021 International Symposium on Physical Design (pp. 15-22).

[5] Dash, A. K., & Adhikari, N. (2024). Unlocking Modern VLSI Placement with Cutting-Edge GPU Acceleration Integrated with Deep Learning Tool Kit. Indian Journal of Science and Technology, 17(44), 4600-4610.

[6] Zhang, H., Ge, Y., Zhao, X., & Wang, J. (2025). Hierarchical deep reinforcement learning for multi-objective integrated circuit physical layout optimization with congestion-aware reward shaping. IEEE Access.

[7] Yang, S., Yang, Z., Li, D., Zhang, Y., Zhang, Z., Song, G., & Hao, J. (2022). Versatile multi-stage graph neural network for circuit representation. Advances in Neural Information Processing Systems, 35, 20313-20324.

[8] Ren, H., Godil, S., Khailany, B., Kirby, R., Liao, H., Nath, S., ... & Roy, R. (2021, November). Optimizing vlsi implementation with reinforcement learning-iccad special session paper. In 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD) (pp. 1-6). IEEE.

[9] Koutlis, D. (2023). Estimation of Voltage Drop in Power Circuits using Machine Learning Algorithms: Investigating potential applications of machine learning methods in power circuits design.
[10] Goldie, A., & Mirhoseini, A. (2020, March). Placement optimization with deep reinforcement learning. In Proceedings of the 2020 International Symposium on Physical Design (pp. 3-7).
[11] Wang, Z., Geng, Z., Tu, Z., Wang, J., Qian, Y., Xu, Z., ... & Wu, F. (2024). Benchmarking end-to-end performance of ai-based chip placement algorithms. arXiv preprint arXiv:2407.15026.
[12] Babu, S. J., Hu, F., Zhu, L., Singhal, S., & Guo, X. (2025). Extending Silicon Lifetime: A Review of Design Techniques for Reliable Integrated Circuits. arXiv preprint arXiv:2503.21165.
[13] Thakoor, S., Tallec, C., Azar, M. G., Munos, R., Veličković, P., & Valko, M. (2021, May). Bootstrapped representation learning on graphs. In ICLR 2021 workshop on geometrical and topological representation learning.
[14] Wang, H., Wang, K., Yang, J., Shen, L., Sun, N., Lee, H. S., & Han, S. (2020, July). GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In 2020 57th ACM/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.
[15] Guo, Z., Liu, M., Gu, J., Zhang, S., Pan, D. Z., & Lin, Y. (2022, July). A timing engine inspired graph neural network model for pre-routing slack prediction. In Proceedings of the 59th ACM/IEEE Design Automation Conference (pp. 1207-1212).
[16] El Sayed, Z., Wang, Z., Selmani, H., Knechtel, J., Sinanoglu, O., & Alrahis, L. (2025). Graph neural networks for integrated circuit design, reliability, and security: Survey and tool. ACM Computing Surveys, 58(4), 1-44.
[17] Du, X., Zhong, R., Kai, S., Tang, Z., Xu, S., Hao, J., ... & Yan, J. (2024, October). JigsawPlanner: Jigsaw-like Floorplanner for Eliminating Whitespace and Overlap among Complex Rectilinear Modules. In Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (pp. 1-9).
[18] Yang, Y., Wang, M., Wang, J., Li, P., & Zhou, M. (2025). Multi-Agent Deep Reinforcement Learning for Integrated Demand Forecasting and Inventory Optimization in Sensor-Enabled Retail Supply Chains. Sensors (Basel, Switzerland), 25(8), 2428.
[19] Xing, S., & Wang, Y. (2025). Cross-Modal Attention Networks for Multi-Modal Anomaly Detection in System Software. IEEE Open Journal of the Computer Society.
[20] Wang, B., Wang, Z., Zhao, W., & Liu, Y. (2025). Network Fabric Simulation and Validation for Data Center Routing Convergence Under Large-Scale Failure Scenarios. Computer Science Bulletin, 8(01), 310-326.
[21] Shen, Z., Wang, Z., & Liu, Y. (2025). Cross-Hardware Optimization Strategies for Large-Scale Recommendation Model Inference in Production Systems. Frontiers in Artificial Intelligence Research, 2(3), 521-540.
[22] Yang, S., Ding, G., & Zeng, Z. (2025). Dynamic Capacity Optimization and Cost Reduction Strategies for Large-Scale Cloud Data Infrastructure. Computer Science Bulletin, 8(01), 290-309.
[23] Xing, S., Wang, Y., & Liu, W. (2025). Self-adapting CPU scheduling for mixed database workloads via hierarchical deep reinforcement learning. Symmetry, 17(7), 1109.
[24] Mai, N. T., Fang, Q., & Cao, W. (2025). Measuring student trust and over-reliance on AI tutors: Implications for STEM learning outcomes. International Journal of Social Sciences and English Literature, 9(12), 11-17.
[25] Han, X., Yang, Y., Chen, J., Wang, M., & Zhou, M. (2025). Symmetry-Aware Credit Risk Modeling: A Deep Learning Framework Exploiting Financial Data Balance and Invariance. Symmetry (20738994), 17(3).
[26] Zeng, Z., Lin, H., Zhang, S., and Wang, B. (2026). Adaptive Robust Watermarking for Large Language Models via Dynamic Token Embedding Perturbation. IEEE Access.

[27] Fang, Q., & Liu, W. (2025). HARLA-ED: Resolving Information Asymmetry and Enhancing Algorithmic Symmetry in Intelligent Educational Assessment via Hybrid Reinforcement Learning. Symmetry, 18(1), 58.

[28] Xing, S., Wang, Y., & Liu, W. (2025). Multi-Dimensional Anomaly Detection and Fault Localization in Microservice Architectures: A Dual-Channel Deep Learning Approach with Causal Inference for Intelligent Sensing. Sensors, 25(11), 3396.

[29] Hu, X., Zhao, X., Wang, J., & Yang, Y. (2025). Information-theoretic multi-scale geometric pre-training for enhanced molecular property prediction. Plos one, 20(10), e0332640.