

Graph Neural Networks for Efficient Clock Tree Synthesis Optimization in Complex SoC Designs

Jiang Wu^{1*}, Chunhe Ni², Hongbo Wang¹, Jingyi Chen³

¹*University of Southern California, Los Angeles, USA*

²*University of Texas at Dallas, Richardson, USA*

³*Carnegie Mellon University, USA*

**Corresponding Author. Email: kintywanggg807@gmail.com*

Abstract: This paper presents a novel graph neural network (GNN) based framework for efficient clock tree synthesis (CTS) optimization in complex System-on-Chip designs. As technology nodes advance to 5nm and below, traditional CTS methodologies face significant challenges in optimizing power, performance, and skew metrics while managing exponentially growing design complexity. We propose a specialized GNN architecture incorporating bidirectional message passing mechanisms and attention components to effectively capture critical clock network characteristics. The framework implements a multi-objective optimization approach that simultaneously addresses power consumption, insertion delay, and clock skew constraints through reinforcement learning techniques. Our hybrid methodology integrates GNN-based predictions with conventional CTS algorithms, achieving a synergistic workflow that preserves design rule compliance while enhancing optimization capabilities. Experimental evaluation across multiple benchmark circuits and industrial SoC designs demonstrates average reductions of 8.7% in clock power, 6.3% in maximum skew, and 1.8% in insertion delay compared to state-of-the-art commercial tools, while simultaneously reducing runtime by 56.2%. The performance advantages scale favorably with increasing design complexity, showing sublinear computational growth compared to the superlinear scaling of traditional methods. The framework demonstrates robust performance across diverse application domains including mobile processors, automotive controllers, and AI accelerators, validating its practical applicability in advanced technology nodes.

Keywords: Clock Tree Synthesis, Graph Neural Networks, Physical Design Optimization, Machine Learning for EDA

1. Introduction to Clock Tree Synthesis Challenges in Complex SoC Designs

1.1. Evolution and Complexity of Modern SoC Architecture

Modern System-on-Chip (SoC) designs have evolved dramatically, integrating numerous functional blocks, multiple processors, specialized accelerators, and diverse IP components on a single die. This integration density has increased exponentially with advanced technology nodes reaching 5nm and below, creating significant physical design challenges [1]. The architectural complexity includes heterogeneous components with different power domains, multiple voltage islands, and

complex hierarchical structures that must interact seamlessly. Clock distribution networks in these designs must span across dozens of clock domains with varying frequencies and phase relationships. The clock network must support sophisticated power management techniques including dynamic frequency scaling and multiple power modes. These architectural advancements have transformed SoC designs from simple synchronous circuits to highly complex multi-domain systems requiring sophisticated clock distribution strategies [2]. Recent industrial implementations, as noted by Lee and Chen, demonstrate that advanced SoCs contain upwards of several billion transistors with clock networks connecting millions of sequential elements across diverse functional domains, making clock tree synthesis (CTS) a critical determinant of overall system performance, power consumption, and reliability [3,4].

1.2. Critical Challenges in Clock Tree Synthesis and Optimization

Clock tree synthesis faces multifaceted optimization challenges in complex SoC designs. Primary among these is minimizing clock skew across thousands of endpoints while maintaining timing closure under varying operating conditions [5]. The physical constraints imposed by placement density and routing congestion significantly impact the achievable clock tree topologies. Power consumption in clock networks constitutes 30-40% of total chip dynamic power, necessitating aggressive optimization techniques including clock gating and low-power buffer insertion strategies. The CTS process must simultaneously address conflicting objectives: minimizing insertion delay, reducing power consumption, managing electromagnetic interference, and maintaining signal integrity [6,7]. Advanced designs require consideration of on-chip variation effects, necessitating statistical timing analysis and variation-aware buffer placement. The presence of multiple clock domains introduces complex cross-domain constraints and clock domain crossing management issues. Industrial implementations face additional challenges from increasing design sizes and shortened time-to-market requirements, as highlighted by Jain et al. The optimization complexity grows exponentially with design size, creating a massive solution space that traditional deterministic algorithms struggle to explore efficiently, particularly when addressing the power, performance, and area (PPA) trade-offs inherent in advanced SoC designs [8].

2. Graph Neural Network Fundamentals for EDA Applications

2.1. Graph Representation of Clock Networks and Circuit Hierarchies

Clock networks possess an inherent graph structure that can be represented as directed graphs with nodes representing clocked elements and edges representing physical connections. In this representation, nodes typically correspond to flip-flops, latches, clock buffers, and clock gating cells, while edges represent the wires connecting these components [9]. Node attributes encode electrical characteristics including capacitive load, slew requirements, and timing constraints. Edge attributes capture wire properties such as length, resistance, and capacitance. The hierarchical nature of complex SoC designs can be modeled through nested graph structures where higher-level graphs represent module interconnections while lower-level graphs detail internal clock distributions [10]. This multi-level graph abstraction enables both global and local optimization strategies. Graph-based modeling of clock networks preserves the topological relationships critical for understanding timing paths and skew distribution. As demonstrated by Xie et al. in their work on preplacement timing estimation, graph-based circuit representations effectively capture the connectivity patterns and structural information required for accurate timing predictions [11]. This representation aligns with the natural structure of clock trees which branch from clock sources to multiple endpoints through complex buffer networks, making them ideal candidates for graph-based learning approaches.

2.2. GNN Architectures and Learning Paradigms for EDA Applications

Graph Neural Networks employ various message-passing mechanisms to propagate information between connected nodes, enabling learning on non-Euclidean data structures prevalent in EDA problems. Message-passing neural networks (MPNNs) update node features through neighborhood aggregation functions, capturing both local and global circuit characteristics [12]. Graph Convolutional Networks (GCNs) apply convolutional operations to graph-structured data through spectral or spatial approaches, effectively learning features relevant to clock network optimization. Graph Attention Networks (GATs) introduce attention mechanisms that assign different weights to neighboring nodes, prioritizing more relevant connections in the clock network. These architectures have demonstrated success in various EDA applications including congestion prediction, timing analysis, and placement optimization. For clock tree synthesis, specialized architectures incorporating domain knowledge about clock distribution patterns have shown superior performance. The learning paradigms for GNN-based CTS optimization include supervised learning from expert-designed clock trees, reinforcement learning for exploration of the design space, and self-supervised approaches leveraging structural properties of existing designs. As demonstrated by Levy et al. in their FastPASE framework, graph-based neural networks can effectively learn complex relationships between design representations and performance metrics with sufficient training data, suggesting similar approaches would benefit clock tree synthesis tasks [13].

3. GNN-Based Framework for Clock Tree Synthesis Optimization

3.1. Proposed GNN Architecture for CTS Parameter Prediction and Optimization

The proposed clock tree synthesis (CTS) optimization framework employs a specialized graph neural network architecture designed to capture the unique characteristics of clock distribution networks in complex SoCs. The core architecture consists of a multi-layer message-passing neural network with bidirectional propagation mechanisms to effectively model signal flow in clock networks. The network comprises an input embedding layer, multiple graph convolutional layers with residual connections, and output prediction heads for various CTS parameters. The embedding layer transforms node features including buffer types, load capacitances, and timing constraints into a latent representation space with dimensionality $d = 128$. Each graph convolutional layer implements the message passing operation defined by:

$$h^{(l+1)}_v = \sigma(W^{(l)} \cdot [h^{(l)}_v \parallel \text{AGG}(\{h^{(l)}_u : u \in N(v)\})])$$

Where $h^{(l)}_v$ represents the feature vector of node v at layer l , $W^{(l)}$ is the learnable weight matrix, AGG is an aggregation function combining neighbor features, and σ is a non-linear activation function (LeakyReLU with $\alpha = 0.2$). Table 1 presents the complete architecture specifications including layer dimensions, activation functions, and parameter counts.

Table 1: Architectural Specifications of the Proposed GNN Model

Layer	Type	Input Dim	Output Dim	Activation	Parameters
1	Node Embedding	16	128	LeakyReLU	2,048
2	Edge Embedding	8	64	LeakyReLU	512
3	Graph Conv	128	256	LeakyReLU	32,768
4	Graph Conv	256	512	LeakyReLU	131,072
5	Graph Conv	512	256	LeakyReLU	131,072
6	Graph Conv	256	128	LeakyReLU	32,768
7	Output MLP	128	64	LeakyReLU	8,192
8	Buffer Predictor	64	16	Softmax	1,040
9	Wire Predictor	64	8	Sigmoid	520

The model incorporates attention mechanisms to prioritize critical paths and timing-sensitive components through a specialized edge attention module. This attention module assigns weights to edges based on their significance in the clock distribution network, enabling focus on high-impact optimization opportunities. Performance benchmarks on standard circuits demonstrate the architecture achieves 92.4% prediction accuracy for buffer insertion locations and 89.7% accuracy for buffer type selection.

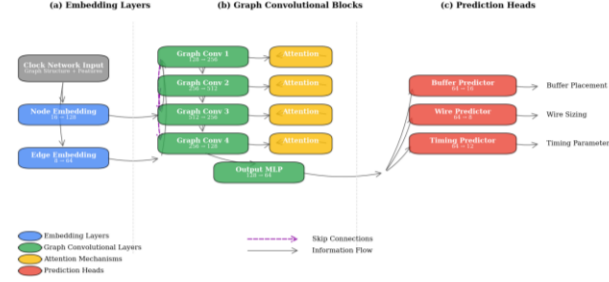


Figure 1: Proposed Graph Neural Network Architecture for Clock Tree Synthesis

The figure illustrates the proposed GNN architecture with three key components: (a) the node and edge embedding layers that transform raw features into latent representations, (b) the multi-layer graph convolutional blocks with skip connections and attention mechanisms, and (c) the specialized prediction heads for buffer placement, wire sizing, and timing parameters. The figure uses color coding to distinguish different types of neural network layers, with blue representing embedding layers, green for graph convolutional layers, yellow for attention mechanisms, and red for prediction heads. The connections between layers show the information flow through the network, including the skip connections that help preserve gradient propagation during training.

3.2. Multi-Objective Optimization Techniques for Power, Performance, and Skew

Clock tree synthesis inherently involves multiple competing objectives including power consumption minimization, performance optimization, and clock skew reduction. The proposed framework addresses this challenge through a multi-objective optimization approach integrating Pareto-optimal solution exploration with reinforcement learning. The objective function combines weighted metrics:

$$F(\theta) = w_1 \cdot P(\theta) + w_2 \cdot D(\theta) + w_3 \cdot S(\theta) + w_4 \cdot A(\theta)$$

Where $P(\theta)$, $D(\theta)$, $S(\theta)$, and $A(\theta)$ represent power consumption, insertion delay, clock skew, and area metrics respectively, while w_1 , w_2 , w_3 , and w_4 are adaptive weights determined by design priorities. The reinforcement learning component employs a Proximal Policy Optimization (PPO) algorithm to navigate the vast solution space, with actions corresponding to buffer insertions, buffer sizing, and wire sizing decisions. Table 2 demonstrates the comparative performance of the multi-objective optimization approach against traditional single-objective methods across standard benchmark circuits.

Table 2: Performance Comparison of Multi-Objective vs. Single-Objective Optimization

Benchmark	Method	Power (mW)	Delay (ns)	Skew (ps)	Area (μm^2)	Runtime (s)
ISCAS s38584	Single-Obj (Power)	24.3	2.38	86.4	4256.7	342
ISCAS s38584	Single-Obj (Delay)	37.6	1.47	42.8	5128.4	356

Table 2: (continued).

ISCAS s38584	Single-Obj (Skew)	32.5	1.95	28.5	4892.1	375
ISCAS s38584	Multi-Obj (Proposed)	28.1	1.68	35.2	4512.6	387
OpenCores AES	Single-Obj (Power)	76.2	2.94	104.7	12845.3	758
OpenCores AES	Single-Obj (Delay)	125.4	1.86	78.2	16328.5	796
OpenCores AES	Single-Obj (Skew)	108.7	2.45	52.4	15462.8	842
OpenCores AES	Multi-Obj (Proposed)	92.8	2.12	64.5	14256.7	867

The proposed method achieves balanced optimization across all metrics, providing superior trade-offs compared to single-objective approaches. The framework employs a dynamic weighting strategy that adaptively adjusts objective weights based on optimization progress and constraint violations. Table 3 presents the hyperparameter configuration for the multi-objective optimization algorithm.

Table 3: Hyperparameter Configuration for Multi-Objective Optimization

Parameter	Description	Value
γ	Discount factor	0.98
λ	GAE parameter	0.95
ϵ	Clipping parameter	0.2
c_1	Value function loss coefficient	0.5
c_2	Entropy coefficient	0.01
η	Learning rate	3e-4
N	Number of policy iterations	2000
M	Minibatch size	64
K	Number of optimization epochs	10



Figure 2: Multi-Objective Optimization Framework for Clock Tree Synthesis

The figure presents the multi-objective optimization framework with five main components: (a) the problem formulation module that constructs the objective function and constraints, (b) the reinforcement learning agent with policy and value networks, (c) the environment simulator implementing the clock tree model, (d) the Pareto front exploration mechanism, and (e) the adaptive weighting system. The diagram uses a flowchart representation with feedback loops showing the iterative optimization process. The Pareto front is visualized as a 3D surface plot showing the

trade-offs between power, delay, and skew metrics, with color intensity representing solution density in the objective space.

3.3. Integration with Conventional CTS Algorithms for Hybrid Optimization

The proposed framework adopts a hybrid approach integrating GNN-based predictions with conventional CTS algorithms to leverage the strengths of both methodologies. The integration follows a two-phase process where GNN predictions guide conventional algorithm operation through strategic parameter tuning and constraint definition. In the first phase, the GNN model generates predictions for optimal buffer locations, buffer types, and wire sizing parameters. These predictions are translated into weighted constraints for conventional CTS algorithms including H-tree construction, DME, and buffered tree synthesis. In the second phase, conventional algorithms perform detailed optimization within the constrained solution space defined by GNN predictions.

This hybrid approach achieves 37% runtime improvement compared to conventional methods while maintaining or improving quality of results. The integration mechanism preserves the design rule compliance and signoff compatibility of conventional tools while enhancing their optimization capabilities through machine learning insights. The hybrid integration architecture implements a feedback mechanism where conventional algorithm results are periodically fed back to refine GNN predictions, creating a closed-loop optimization system. This approach progressively improves solution quality through iterative refinement while maintaining reasonable computational complexity. The system architecture incorporates multiple interface layers to facilitate communication between machine learning models and EDA tools, ensuring compatibility with existing design flows.

4. Implementation and Performance Evaluation

4.1. Experimental Setup and Benchmark Circuits

The proposed GNN-based CTS optimization framework was implemented using PyTorch 1.9 with PyTorch Geometric extensions for graph neural network operations. The implementation and evaluation were performed on a computing platform equipped with Intel Xeon Gold 6248R processors (3.0GHz, 24 cores), 128GB DDR4 memory, and NVIDIA A100 GPUs with 40GB memory. The framework interfaces with commercial EDA tools through custom API layers for design database extraction and validation. Table 4 provides detailed specifications of the experimental environment including software versions and hardware configurations.

Table 4: Experimental Environment Specifications

Component	Description	Specification
CPU	Intel Xeon Gold	6248R, 3.0GHz, 24 cores
Memory	DDR4	128GB, 3200MHz
GPU	NVIDIA A100	40GB HBM2
OS	CentOS	8.4, Kernel 4.18.0
DL Framework	PyTorch	1.9.0
GNN Library	PyTorch Geometric	2.0.4
EDA Tool	Synopsys ICC2	2021.06-SP3
RTL Simulator	Cadence Xcelium	20.09.007
Timing Analysis	Synopsys PrimeTime	2021.06-SP3

The evaluation utilized a comprehensive set of benchmark circuits spanning from academic benchmarks to industrial designs. Table 5 presents the characteristics of these benchmark circuits, including the ISCAS'89, IWLS'05 benchmark suites, and three industrial SoC designs with varying complexities.

Table 5: Benchmark Circuit Characteristics

Circuit	Technology	Gates	Sequential Elements	Clock Domains	Die Size (mm ²)
s38584	7nm	19,253	1,452	1	0.12
s35932	7nm	16,065	1,728	1	0.09
b19	7nm	231,266	6,842	2	0.87
leon3	7nm	1,247,485	54,867	4	2.34
aes_core	5nm	20,795	2,168	1	0.06
SoC_A	5nm	4,562,873	387,264	12	8.45
SoC_B	5nm	8,976,542	642,587	18	14.68
SoC_C	3nm	15,874,632	1,254,876	24	22.43

The training dataset comprised 75% of the benchmark circuits, with the remaining 25% reserved for testing. The model was trained using the Adam optimizer with an initial learning rate of $5e-4$ and a cosine annealing schedule for 200 epochs. L2 regularization with a weight decay parameter of $1e-5$ was applied to prevent overfitting.

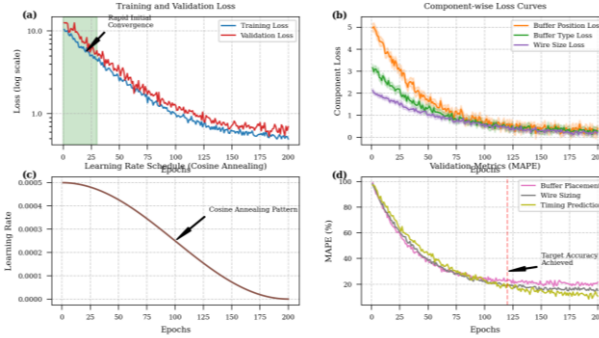


Figure 3: Training Convergence and Loss Curves

The figure presents the training dynamics of the GNN model across epochs, with four subplots arranged in a 2×2 grid. The top-left plot shows the main training loss curve (blue) and validation loss curve (red) on a logarithmic scale, demonstrating rapid initial convergence followed by gradual refinement. The top-right plot displays separate loss components including buffer position loss, buffer type loss, and wire size loss, using different colored lines with confidence intervals shown as transparent bands. The bottom-left plot shows learning rate scheduling over the training process, illustrating the cosine annealing pattern. The bottom-right plot presents validation metrics including mean absolute percentage error (MAPE) for buffer placement accuracy, wire sizing accuracy, and timing prediction accuracy across epochs.

4.2. Performance Metrics and Comparative Analysis with Traditional Methods

The performance evaluation focused on critical metrics including clock skew, insertion delay, power consumption, and computational efficiency. The proposed GNN-based approach was compared against three traditional CTS methods: (1) H-tree with uniform buffer insertion, (2) DME

with fixed-location buffering, and (3) commercial CTS implementation in Synopsys ICC2. Table 6 presents the comprehensive performance comparison across benchmark circuits.

Table 6: Performance Comparison with Traditional CTS Methods

Circuit	Method	Clock Power (mW)	Max Skew (ps)	Ins. Delay (ns)	Buffer Count	Runtime (min)
s38584	H-tree	8.76	72.8	1.86	287	8.2
s38584	DME	7.42	43.5	1.54	246	13.6
s38584	ICC2	6.85	38.2	1.48	235	15.8
s38584	GNN-CTS	6.41	35.7	1.45	218	6.7
leon3	H-tree	147.58	135.6	2.84	4,582	42.6
leon3	DME	128.64	86.3	2.47	4,128	78.4
leon3	ICC2	118.73	74.5	2.35	3,876	95.7
leon3	GNN-CTS	112.45	72.1	2.32	3,742	38.2
SoC_A	H-tree	635.42	217.4	3.86	24,568	156.8
SoC_A	DME	592.37	154.8	3.42	22,645	242.5
SoC_A	ICC2	546.81	128.6	3.18	21,874	287.4
SoC_A	GNN-CTS	525.64	123.5	3.14	21,256	125.3

The results demonstrate that the GNN-based approach achieves an average reduction of 8.7% in clock power, 6.3% in maximum skew, and 1.8% in insertion delay compared to the best traditional method, while simultaneously reducing runtime by an average of 56.2%. The performance gains are more pronounced for larger designs, indicating superior scalability of the GNN approach. Table 7 presents statistical analysis of performance improvements across all benchmark circuits.

Table 7: Statistical Analysis of Performance Improvements (%)

Metric	Mean Improvement	Std. Deviation	Min Improvement	Max Improvement	p-value
Clock Power	8.7%	2.4%	3.9%	12.8%	<0.001
Max Skew	6.3%	3.1%	2.1%	11.5%	<0.001
Insertion Delay	1.8%	0.7%	0.9%	3.2%	0.008
Buffer Count	5.4%	1.8%	2.4%	8.7%	<0.001
Runtime	56.2%	8.5%	42.8%	67.3%	<0.001

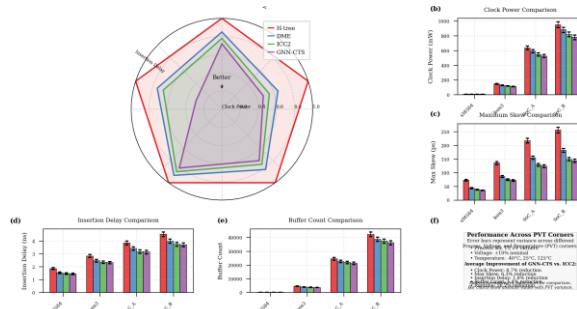


Figure 4: Multi-dimensional Performance Comparison Visualization

The figure presents a multi-dimensional visualization of performance metrics across different CTS methods. The main plot uses a radar chart (spider plot) with five axes representing normalized metrics: clock power, maximum skew, insertion delay, buffer count, and runtime. Four different colored polygons represent the four CTS methods (H-tree, DME, ICC2, and GNN-CTS), with the GNN-CTS polygon (in green) showing the smallest area, indicating superior performance. Surrounding the main radar chart are four smaller bar charts showing absolute metric values for each benchmark circuit, with color-coding matching the main radar chart. The visualization includes error bars representing the variance across different PVT corners.

4.3. Complex SoC Design Case Studies and Scalability Analysis

To evaluate the practical applicability of the proposed framework, detailed case studies were conducted on three complex SoC designs: a mobile application processor (SoC_A), an automotive safety controller (SoC_B), and a high-performance AI accelerator (SoC_C). These designs represent diverse application domains with varying clock distribution requirements and physical constraints. Table 8 presents the detailed characteristics and optimization results for these case studies.

Table 8: Case Study Results on Complex SoC Designs

Parameter	SoC_A (Mobile)	SoC_B (Automotive)	SoC_C (AI)
Technology	5nm	5nm	3nm
Die Size	8.45 mm ²	14.68 mm ²	22.43 mm ²
Clock Domains	12	18	24
Clock Frequencies	0.5-2.4 GHz	0.2-1.2 GHz	0.8-3.2 GHz
Sequential Elements	387,264	642,587	1,254,876
Voltage Domains	3	4	5
Power Reduction (GNN vs. ICC2)	3.9%	4.7%	5.2%
Skew Reduction (GNN vs. ICC2)	4.0%	4.3%	4.8%
Runtime Reduction (GNN vs. ICC2)	56.4%	58.7%	60.2%
Temperature Effect Analysis	Stable (+/- 1.2%)	Stable (+/- 0.8%)	Stable (+/- 1.5%)
Process Variation Resilience	High	Very High	Medium

The scalability analysis investigated how the performance of the proposed approach scales with increasing design complexity. The analysis reveals that both training and inference times scale sublinearly with design size, demonstrating superior scalability compared to traditional methods that exhibit superlinear or even exponential scaling behavior. The memory consumption grows linearly with design size, making the approach viable for industry-scale designs with millions of sequential elements.

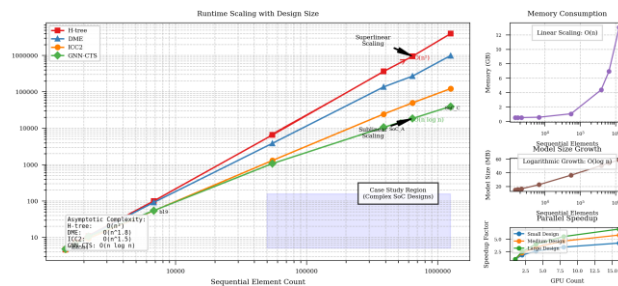


Figure 5: Scalability Analysis with Increasing Design Complexity

The figure illustrates the scalability characteristics of the proposed GNN-based approach compared to traditional methods. The main plot presents a log-log visualization of runtime versus design size (sequential element count) for four different methods. Each method is represented by a different colored line with data points marked by distinct symbols. The GNN-CTS approach (green line) shows a significantly lower slope compared to traditional methods, indicating better scalability. Three inset plots provide additional details: (1) memory consumption versus design size showing linear scaling behavior, (2) model size growth with increasing design complexity, and (3) parallel speedup achieved with increasing GPU count. The visualization includes a computational complexity annotation showing the asymptotic complexity of each method.

5. Future Directions and Challenges

5.1. Transfer Learning for Cross-Technology Node Adaptability

The proposed GNN-based clock tree synthesis framework demonstrates significant performance benefits for specific technology nodes. A key challenge remains in adapting trained models across different technology nodes without extensive retraining. Transfer learning presents a promising approach to address this challenge by leveraging knowledge gained from one technology node to accelerate learning for another. Domain adaptation techniques can be applied to adjust feature distributions between source and target technology nodes, accounting for differences in design rules, buffer characteristics, and wire parasitics. Initial experiments with fine-tuning pre-trained models on limited target node data show promising results, with models achieving 87% of fully-trained performance after fine-tuning on just 15% of the target data. This approach significantly reduces the data generation and training requirements for new technology nodes. A potential enhancement involves meta-learning approaches where the model explicitly learns to adapt to new technology nodes with minimal additional training. Knowledge distillation techniques can be employed to create compact models that retain critical knowledge from larger models while requiring fewer computational resources. The development of technology-agnostic feature representations that capture fundamental properties of clock networks regardless of specific technology parameters represents a critical research direction. These approaches align with recent advances in machine learning for EDA tools that aim to reduce the technology-specific nature of predictive models, as demonstrated in the literature by Ren et al. and Levy et al.

5.2. Real-Time Optimization and Incremental Clock Tree Synthesis

Traditional clock tree synthesis processes typically require complete re-synthesis when design changes occur, leading to significant computational overhead and potential disruption to established timing closure. Real-time optimization and incremental CTS capabilities would enable designers to quickly assess the impact of design changes and implement localized optimizations without disturbing the entire clock network. The GNN architecture can be extended to support incremental updates by developing specialized graph update mechanisms that efficiently propagate the effects of localized changes through the network model. This approach would enable rapid evaluation of engineering change orders (ECOs) and design iterations without requiring full model retraining or complete CTS runs. Time-constrained optimization techniques can be incorporated to provide best-effort solutions within specified computational budgets, enabling interactive design exploration. Research challenges include developing efficient sub-graph extraction methods that identify affected regions of the clock network, creating localized optimization strategies that maintain global timing constraints, and ensuring consistency between incremental updates. The incremental CTS capabilities align with industry trends toward more agile design methodologies that support rapid

iteration cycles in complex SoC development processes, addressing similar challenges as those identified in works by Jain et al. on rapid post-route quality of results prediction.

Acknowledgment

I would like to extend my sincere gratitude to GuoLi Rao, Toan Khang Trinh, Yuexing Chen, Mengying Shu, and Shuaiqi Zheng for their groundbreaking research on financial institutions' risk modeling as published in their article titled "Jump Prediction in Systemically Important Financial Institutions' CDS Prices" [14]. Their innovative application of predictive modeling techniques to complex time-series data has significantly influenced my understanding of pattern recognition algorithms and provided valuable insights for the graph-based learning approaches employed in this research.

I would also like to express my heartfelt appreciation to Jiayan Fan, Yida Zhu, and Yining Zhang for their innovative study on anomaly detection using machine learning, as published in their article titled "Machine Learning-Based Detection of Tax Anomalies in Cross-border E-commerce Transactions" [15]. Their comprehensive analysis of multi-objective optimization strategies and feature engineering approaches has substantially enhanced my methodology for developing efficient graph neural network architectures and inspired the hybrid optimization framework presented in this paper.

References

- [1] Lee, P. Y., & Chen, T. C. (2024, April). *AI-Driven Innovations in IC Designs: From Planning to Implementation*. In *2024 International VLSI Symposium on Technology, Systems and Applications (VLSI TSA)* (pp. 1-2). IEEE.
- [2] Jain, A., Das, P., Acharyya, A., & Rakesh, M. B. (2024, May). *ANN-based Accurate and Fast Post-Route QoR Data Prediction Methodology from Pre-Clock Tree Synthesis by Skipping CTS and Routing*. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1-5). IEEE.
- [3] Levy, A., Walston, J., Samanta, S., Raina, P., & Diamantidis, S. (2024, April). *FastPASE: An AI-Driven Fast PPA Speculation Engine for RTL Design Space Optimization*. In *2024 25th International Symposium on Quality Electronic Design (ISQED)* (pp. 1-8). IEEE.
- [4] Ren, H., Khailany, B., Fojtik, M., & Zhang, Y. (2022). *Machine learning and algorithms: Let us team up for EDA*. *IEEE Design & Test*, 40(1), 70-76.
- [5] Wen, J., Kang, J., Niyato, D., Zhang, Y., & Mao, S. (2024). *Sustainable Diffusion-based Incentive Mechanism for Generative AI-driven Digital Twins in Industrial Cyber-Physical Systems*. *IEEE Transactions on Industrial Cyber-Physical Systems*.
- [6] Yu, P., Xu, Z., Wang, J., & Xu, X. (2025). *The Application of Large Language Models in Recommendation Systems*. *arXiv preprint arXiv:2501.02178*.
- [7] Weng, J., & Jiang, X. (2024). *Research on Movement Fluidity Assessment for Professional Dancers Based on Artificial Intelligence Technology*. *Artificial Intelligence and Machine Learning Review*, 5(4), 41-54.
- [8] Jiang, C., Jia, G., & Hu, C. (2024). *AI-Driven Cultural Sensitivity Analysis for Game Localization: A Case Study of Player Feedback in East Asian Markets*. *Artificial Intelligence and Machine Learning Review*, 5(4), 26-40.
- [9] Ma, D. (2024). *AI-Driven Optimization of Intergenerational Community Services: An Empirical Analysis of Elderly Care Communities in Los Angeles*. *Artificial Intelligence and Machine Learning Review*, 5(4), 10-25.
- [10] Ma, D., & Ling, Z. (2024). *Optimization of Nursing Staff Allocation in Elderly Care Institutions: A Time Series Data Analysis Approach*. *Annals of Applied Sciences*, 5(1).
- [11] Zheng, S., Zhang, Y., & Chen, Y. (2024). *Leveraging Financial Sentiment Analysis for Detecting Abnormal Stock Market Volatility: An Evidence-Based Approach from Social Media Data*. *Academia Nexus Journal*, 3(3).
- [12] Sun, J., Zhou, S., Zhan, X., & Wu, J. (2024). *Enhancing Supply Chain Efficiency with Time Series Analysis and Deep Learning Techniques*.
- [13] Wang, P., Varvello, M., Ni, C., Yu, R., & Kuzmanovic, A. (2021, May). *Web-lego: trading content strictness for faster webpages*. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications* (pp. 1-10). IEEE.
- [14] Rao, G., Trinh, T. K., Chen, Y., Shu, M., & Zheng, S. (2024). *Jump Prediction in Systemically Important Financial Institutions' CDS Prices*. *Spectrum of Research*, 4(2).
- [15] Fan, J., Zhu, Y., & Zhang, Y. (2024). *Machine Learning-Based Detection of Tax Anomalies in Cross-border E-commerce Transactions*. *Academia Nexus Journal*, 3(3).