Differentiable Graph Neural Networks for Wirelength Estimation

Zhengfeng Wu, Pratik Shrestha, Saran Phatharodom, and Ioannis Savidis
Department of Electrical and Computer Engineering
Drexel University
Philadelphia, PA

2025 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3503-5683-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISCAS56072.2025.11043324

***Abstract*—A model that utilizes a graph neural network (GNN) is proposed to estimate wirelength in the early stages of physical design. The model predicts the post-routing wirelength of a net, which addresses the limitations of current estimators including half-perimeter wirelength (HPWL) that tend to under-estimate the true post-routed wirelength of complex nets. Utilizing a dataset of 18 benchmark circuits, the GNN achieves an average R^2 of 0.825, outperforming HPWL, which yields an R^2 of 0.764. The GNN demonstrates consistent performance across nets of varying lengths, providing significantly improved accuracy over HPWL for long and multi-path nets. The GNN model is differentiable and compatible with gradient-based optimization methods, while providing an 8× improvement in inference time.**

I. Introduction and Background

Accurate wirelength estimation in the early stages of physical design plays a critical role in the final power, performance, and area (PPA) of a circuit. Key metrics that are affected include timing, signal integrity, power consumption, and routability. The digital synthesis flow, which is shown in Fig. 1, includes multiple stages that are contingent on accurate estimates of wirelength. In floorplanning, early estimates of wirelength guide the placement of functional blocks to minimize congestion and optimize signal timing. Placement determines exact cell locations, with estimates of wirelength driving optimization under timing and congestion constraints.

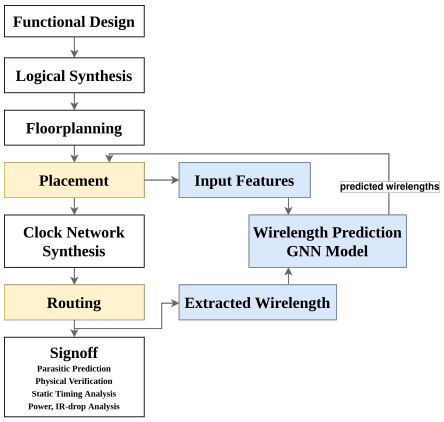


Fig. 1: Synthesis and physical design flow of a digital circuit. A GNN-based wirelength estimator is developed to guide placement.

Many techniques have been proposed to estimate interconnect lengths during the early stages of physical design [1]–[5]. Among prior techniques, half-perimeter wirelength (HPWL) is most frequently used as a wirelength estimator that provides high computational efficiency. HPWL calculates the half-perimeter of the bounding box that encloses the pins i of a given net k, and is calculated as

$$\text{HPWL}_k = \left(\max_{i\in\mathcal{P}_k} x_i - \min_{i\in\mathcal{P}_k} x_i\right) + \left(\max_{i\in\mathcal{P}_k} y_i - \min_{i\in\mathcal{P}_k} y_i\right), \quad (1)$$

where $\mathcal{P}_k$ represents the set of pins of net k, and x_i and y_i represent the x- and y-coordinates of the pins $i \in \mathcal{P}_k$, respectively.

More advanced methods such as rectilinear Steiner minimal tree (RSMT) [2] and fast lookup table estimation (FLUTE) [3] provide greater accuracy, but at the cost of greater computational resources. The RSMT algorithm [2] involves constructing a tree by selecting specific edges and paths between points (e.g., pins of a net) that minimize the overall wirelength of a rectilinear grid. The FLUTE wirelength estimator [3] utilizes discrete lookup tables to approximate the RSMT for a given net.

Gradient-based optimization algorithms used in circuit placement require a differentiable objective function (*i.e.*, wirelength). However, neither FLUTE [3] nor RSMT [2] is differentiable, with both methods potentially resulting in sudden, non-continuous changes in the estimate of wirelength when the positions of the pins change even slightly. Calculation of the HPWL, which involves max and min operations, is inherently non-differentiable. Various approaches have been developed that provide differentiable approximations of the HPWL, smoothing the discontinuities at the non-differentiable points. Techniques that approximate HPWL include the weighted average function [6] and the log-sum-exp function [7].

However, errors in the approximation of HPWL are introduced with such methods, resulting in a trade-off between differentiability and accuracy. In addition, HPWL is an approximation of the true wirelength. Therefore, utilizing an approximation of the approximated wirelength (*i.e.*, HPWL) results in deviation from the true wirelength, which impacts the overall accuracy of the placement algorithms.

Machine learning has been increasingly applied to estimate the wirelength of circuit nets [8]–[11]. Machine learning models allow for the prediction of any metric for which sufficient data is available. Instead of setting HPWL, which is an approximation of wirelength, as the learning target, a more accurate approach is to learn the true wirelength extracted after routing.

Prior work has focused on the development of ML-based estimators of total wirelength [8], [9]. Linear discriminant analysis (LDA) is applied to predict per-path wirelength in timing-critical paths after virtual placement [10]. More recently, a per-net wirelength estimator using graph attention networks was proposed [11], which provides finer-grained

predictions that are critical for precise timing estimates and congestion management.

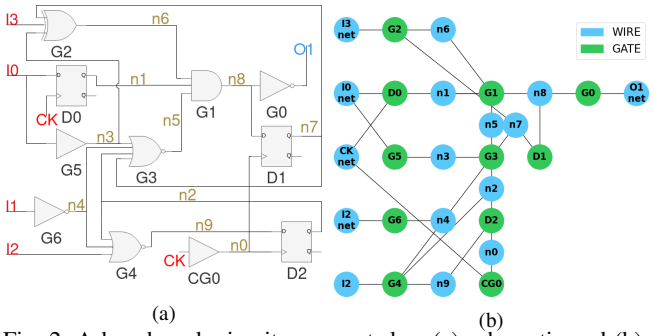
In this work, a differentiable model that utilizes a graph neural network (GNN) is proposed to estimate the wirelength of each interconnect of a circuit in early physical design stages. The proposed GNN model provides a more accurate estimate of the true routing length than HPWL while performing placement. In addition, the differentiability of neural networks allows for use with gradient-based optimization algorithms for placement and routing.

The remainder of the paper is structured as follows. In Section II, the formulation of the wirelength estimation problem is provided. In addition, the proposed GNN model that estimates wirelength is introduced. The simulation framework and evaluation of the developed model on benchmark circuits are presented in Section III. A detailed discussion of the accuracy and efficiency of the proposed model, as compared to current estimators, is provided in Section IV. Finally, some concluding remarks are offered in Section V.

II. Differentiable Heterogeneous Graph Neural Network to Approximate the True Post-Routing Wirelength

Graph neural networks (GNNs) have gained significant attention in the modeling of integrated circuits [11]–[14], where the circuit is represented in the form of a graph. A GNN model utilizes the topological structure of the circuit as a primary feature for learning. Specifically, the feature embeddings of each node in the graph are iteratively updated by aggregating information from neighboring nodes, which captures the local structural relationships of the circuit.

In this work, a gate-level digital circuit is represented as a heterogeneous graph that is comprised of two distinct node types: *gate* and *net*. For a given circuit placement, the feature set of the *gate* nodes includes the fan-in, fan-out, width, and coordinates of the gate cells. For *net* nodes, features include the number of gates connected to the output of a driving gate (output degree) and the coordinates of the pins on a net. A benchmark circuit and the corresponding graph representation are shown in Fig. 2.



(a) (b)

Fig. 2: A benchmark circuit represented as (a) schematic and (b) a netlist graph.

A heterogeneous graph neural network is proposed that consists of a relational graph convolution network (RGCN) [15], which is followed by a multi-layer perceptron (MLP)-based predictor for final scoring. The RGCN utilizes GraphSAGE convolution [16]. An initial linear transformation is applied to independently map the features of each node type. Specifically, for *gate* nodes and *net* nodes, the transformation is given by, respectively,

$$h_0^{\text{gate}} = W_{\text{gate}} x^{\text{gate}} \text{ and } h_0^{\text{net}} = W_{\text{net}} x^{\text{net}}, \tag{2}$$

where W_{gate} and W_{net} are weight matrices specific to each node type, and x^{gate} and x^{net} represent the input features of each node type.

The model then performs a GraphSAGE-based message passing on heterogeneous nodes, where each node aggregates information from neighboring nodes that are of a different type. For each *net* node i, the GraphSAGE operation aggregates messages from the set of neighboring *gate* nodes N_i connected by i, using a mean aggregator as given by

$$h_i^{l+1} = \sigma\left(W_{gnn} \cdot \text{concat}\left(h_i^l, \frac{1}{|N_i|}\sum_{j \in N_i} h_j^l\right)\right), \tag{3}$$

where σ is the Leaky ReLU activation, W_{gnn} is the weight matrix for the GNN layer, and h_i^l represents the embedding of node i at layer l. The operation is fully differentiable, which allows for the calculation of the gradient through each GraphSAGE layer by leveraging matrix multiplications, concatenation, and non-linear activations.

The updated node embeddings from the GNN layers are then applied to an MLP-based predictor, which processes the *net* node embeddings through a series of n fully connected layers. Each layer k in the MLP applies a transformation with a weight matrix W_k and a Leaky ReLU activation. For the final MLP layer, a scalar score is provided as the output of the model, which is given by

$$h_{\text{MLP}}^{(k)} = \sigma(W_k h_{\text{MLP}}^{(k-1)}), \quad k = 1, 2, \ldots, n \tag{4}$$

where $W_1, W_2, \ldots, W_n$ are weight matrices for each layer of the MLP.

Differentiability across all GNN layers and the MLP predictor ensures that the model is amenable to gradient-based optimization techniques. During model training, the objective is to minimize the loss function such that the parameters of the GNN are optimized. For post-training placement optimization, the objective shifts to optimizing input variables, specifically placement coordinates, to minimize the output of the GNN model (*i.e.,* predicted wirelength).

III. Simulation Results of Estimating Post-Routing Wirelength with the GNN Model

The 18 IWLS'05 sequential benchmark circuits [17] used to generate a dataset of layouts on a production 28 nm technology node are listed in Table I. The circuits are synthesized into a gate-level netlist using Synopsys Design Compiler, and then placed and routed with Synopsys IC Compiler. The circuits are designed to operate at a frequency of 1 GHz, provide an aspect ratio of 0.5, and utilize 70% of the floorplan area.

The netlist is parsed, circuit features are extracted, and the resulting data is structured, using EDA-schema [18]. Pin coordinates from the netlist produced after placement are used to calculate the HPWL. The actual wirelength of each net is obtained from the post-routed layout. The variation between the post-placement pin coordinates (red circles) and the post-routing pin coordinates (blue squares) of a net is shown in

Fig. 3. The HPWL is represented by the red dotted bounding box, while the true wirelength of a routed net is represented by the blue line connecting the post-routing pin coordinates. The HPWL calculated post-placement is used as a baseline to evaluate the performance of the GNN model in predicting post-routing wirelength.

TABLE I: Number of inputs, outputs, gates, and flip-flops of the 18 benchmark circuits utilized as a dataset for wirelength prediction.

Circuits	No. of inputs	No. of outputs	No. of gates	No. of flip-flops
aes_core	259	129	12575	12186
des3_area	240	64	1921	1784
des3_perf	234	64	51969	50633
ethernet	96	115	23434	21886
fpu	70	40	31436	30895
mem_ctrl	115	152	3978	3774
pci	162	207	7818	7529
simple_spi	16	12	382	373
spi	47	45	1303	1225
ss_pcm	19	9	176	164
systemcaes	260	129	4440	4235
systemcdes	132	65	1953	1860
tv80	14	32	4259	4214
usb_funct	128	121	6666	7099
usb_phy	15	18	289	274
vga_lcd	89	109	32431	31009
wb_conmax	1130	1416	19896	17382
wb_dma	217	215	2478	2188

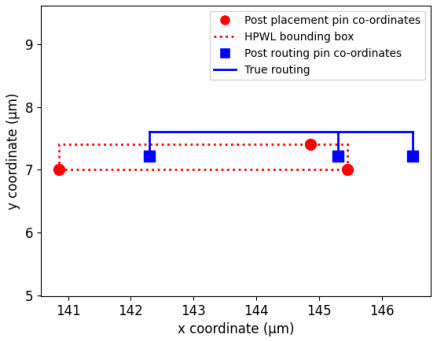


Fig. 3: Comparison of post-placement and post-routing pin coordinates, HPWL bounding box, and the true route of a net.

Cross-validation is performed on a dataset of 18 benchmark circuits, where a single circuit is selected for inference while the remaining 17 circuits are used for training during each executed run. After hyperparameter tuning, the GNN consists of one GraphSAGE layer and six post-processing linear layers. The wirelength predicted by the GNN is plotted against the true wirelength of each routed net, as shown in Fig. 4, with 45° dashed lines included in the sub-figures to delineate the ideal predicted value. For each circuit where the GNN model predicted the wirelength, a corresponding figure is shown with the predicted HPWL wirelength plotted against the true routed wirelength. The R^2 score for either the HPWL-based or GNN-based prediction of wirelength is also provided in the corresponding sub-figure. The calculated per-circuit average and lumped average R^2 score and mean absolute error (MAE) are listed in Table II. The per-circuit average reflects the mean of a given metric (i.e. R^2, MAE, etc.) calculated separately for each circuit and then averaged, while the lumped average combines all circuit data into a single set, and then averages the calculated combined metric by dividing by the total number of nets.

Note that weighted average [6] and log-sum-exp [7] are also evaluated on the 18 benchmark circuits. As the weighted average and log-sum-exp approximators of HPWL result in a near-perfect match of the HPWL, the predictions provided by the HPWL approximators are not included. Therefore, this work provides a comparison between the calculated HPWL and the predicted wirelength returned by the GNN model during placement when estimating the post-routed true wirelength.

TABLE II: Comparison of R^2, MAE, and execution time between HPWL and the GNN model that predicts wirelength.

	HPWL	GNN	Difference(%)
Lumped Avg. R^2	0.675	0.740	+9.6%
Avg. R^2	0.764	0.825	+8.0%
Lumped Avg. MAE (um)	7.44	8.51	+14.4%
Avg. MAE (um)	5.87	1.41	-76.0%
Top 10% R^2	0.614	0.711	+15.8%
Top 5% R^2	0.555	0.703	+26.7%
Top 1% R^2	0.235	0.712	+203.0%
Avg. training time (s)	N/A	282.6	N/A
Avg. inference time (s)	0.471	0.058	-87.7%

IV. Analysis of Results

There is a variance in the prediction across different circuits provided by either HPWL or GNN, as indicated by results shown in Fig. 4. For longer routes, HPWL tends to underestimate the true routing length, which results in a lower R^2 score and a higher MAE. Factors that result in a difference between the HPWL and the post-routed true wirelength include congestion, pin density, the version of the executed placement and routing algorithms, and the specific design constraints. HPWL is less accurate when calculated for nets routed in congested or highly constrained areas of the circuit, which often require rerouted paths.

As indicated by the results listed in Table II, the GNN model provides a lumped total R^2 score of 0.740 and an average R^2 score across all circuits of 0.825, while HPWL results in a lumped total R^2 score of 0.675 and an average R^2 score of 0.764. The results indicate that the GNN provides improved estimates of the post-routing wirelength. The R^2 scores of GNN-predicted wirelength are compared against the R^2 scores of the HPWL analytical model for each of the 18 benchmark circuits, with results shown in Fig. 5. From the comparison of R^2 scores, the GNN outperforms the calculated HPWL in estimating the post-routing wirelength of each net. As indicated by the results listed in Table II, the GNN provides an average per-circuit improvement of 8.0% in the predicted wirelength.

In addition, the R^2 score listed in Table II that is calculated between the computed HPWL and the true routing length for the longest 10%, 5%, and 1% of the nets of each benchmark circuit (sorted based on post-routing true lengths) is 0.614, 0.555, and 0.235, respectively, which indicates a degradation in the prediction of the wirelength of longer nets. In comparison, the GNN returns predicted wirelengths with less deviation from the true wirelengths of routed nets across the 10%, 5%, and 1% longest nets of a circuit with R^2 scores of 0.711, 0.703, and 0.712, respectively. The results, therefore, indicate that the GNN provides consistent performance when predicting the wirelength of long nets. Notably, the GNN provides a 203.0% improvement in R^2 score for the 1% longest nets.

As listed in Table II, training GNNs incurs a fixed cost of 282.6 seconds in CPU time. However, once trained, as the results in Table II indicate, the GNN inference time is 87.7% shorter than the time required to evaluate the HPWL, which is equivalent to an 8.1× speedup. Note that in practice, the

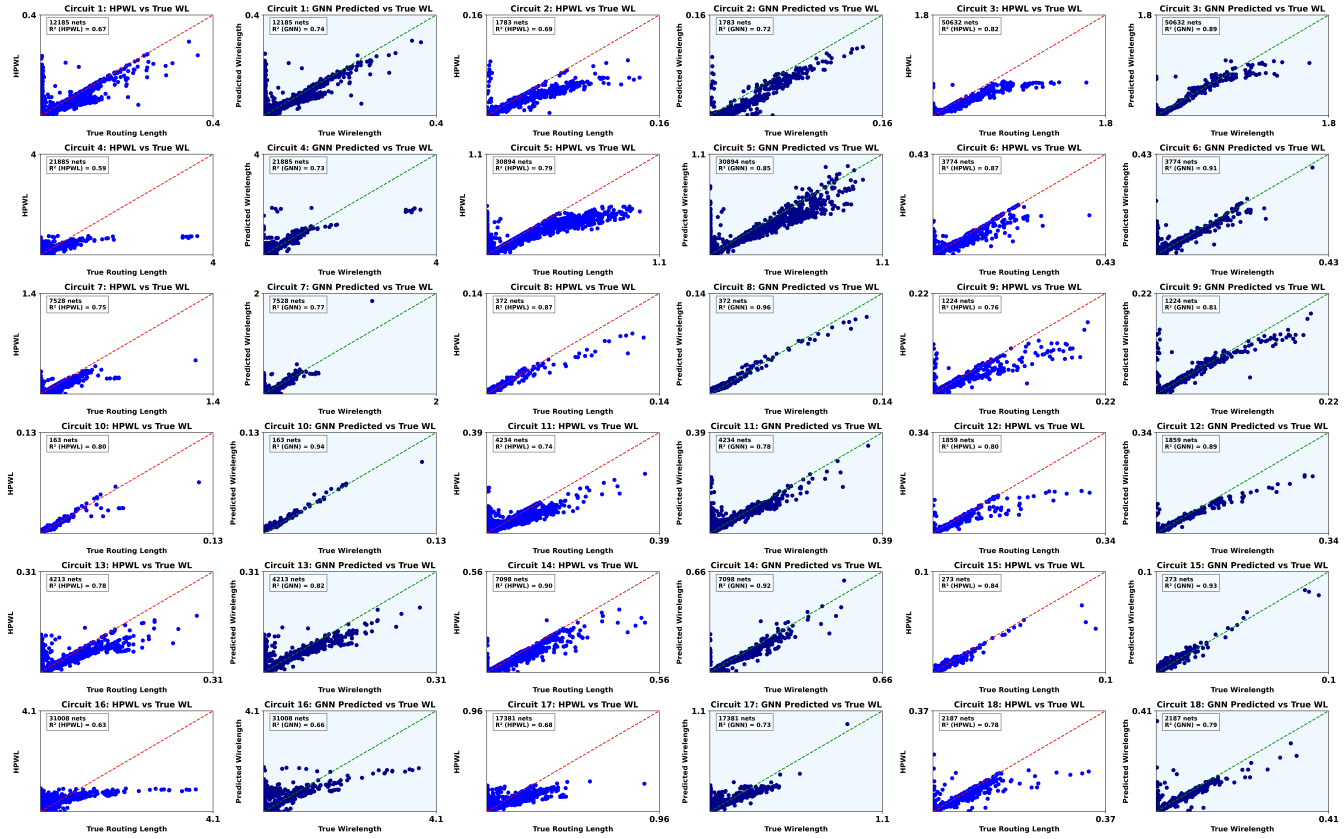


Fig. 4: Comparison of HPWL and GNN predicted wirelength with the true routing lengths of the 18 benchmark circuits analyzed. For each set of two columns, the left column represents the results of the calculated HPWL with respect to the true routing length, while the right column represents the results of the GNN-predicted wirelength with respect to the true routing length. The 45° dashed line in each sub-plot represents an ideal perfect prediction. The number of nets and the R^2 score of the calculated HPWL and the GNN predicted wirelength are annotated on each subplot. The units of the x- and y-axis are in millimeters.

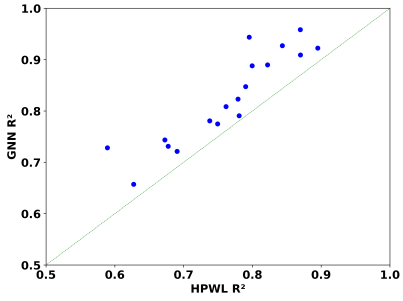


Fig. 5: A comparison of the R^2 score of the GNN model and the R^2 score of the calculated HPWL with respect to the post-routed true wirelength of the 18 benchmark circuits.

inference time of a GNN is contingent on the architecture and size of the implemented model, which is determined by the size of the available training set.

GNNs learn from all of the available features, accounting for factors beyond the physical geometry of the pins utilized to calculate HPWL. For example, the GNN adjusts the predicted wirelength based on the degree of a *net* node. Therefore, the GNN results in more accurate and consistent predictions of wirelength.

Despite the advantages of improved accuracy, faster execution, and differentiability, challenges remain in using GNNs as surrogate objective functions for wirelength in placement optimization. One of the key challenges is a vanishing or exploding gradient [19]. In cases where gradients become excessively small or large during backpropagation, the ability of the neural network to update weights effectively is compromised, which potentially results in inefficient training or even divergence. Another potential challenge is that neural networks are inherently non-convex due to the complex, layered architecture of the network, which introduces multiple local minima or maximum when utilized for optimization [20]. In placement optimization, the non-convexity potentially results in suboptimal estimation of wirelength if the network converges to a poor local minimum.

V. Conclusions

The proposed GNN model that estimates wirelength outperforms the traditional estimate provided by the calculated HPWL, particularly for longer nets. The GNN achieves an average per-circuit R^2 of 0.825, as compared to an R^2 of 0.764 for HPWL. In addition, the GNN provides more consistent estimates across routes with different lengths, providing an R^2 of 0.712 for the 1% of longest nets as compared to an R^2 of 0.235 provided by HPWL. Once trained, the GNN model provides 8.1× faster inference than HPWL, making the proposed GNN model a viable candidate for integration into placement optimization frameworks that utilize estimates of wirelength.

REFERENCES

[1] X. Yang, E. Bozorgzadeh, and M. Sarrafzadeh, "Wirelength estimation based on rent exponents of partitioning and placement," *Proceedings of the International Workshop on System-level Interconnect Prediction(SLIP)*, pp. 25–31, Mar. 2001.

[2] C. Chu and Y.-C. Wong, "Fast and accurate rectilinear Steiner minimal tree algorithm for VLSI design," *Proceedings of the International Symposium on Physical Design(ISPD)*, pp. 28–35, Apr. 2005.

[3] C. Chu and Y. Wong, "FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, No. 1, pp. 70–83, Jan. 2008.

[4] P. Liao, H. Liu, Y. Lin, B. Yu, and M. Wong, "On a moreau envelope wirelength model for analytical global placement," *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, Jul. 2023.

[5] B. N. B. Ray and S. Balachandran, "An efficient wirelength model for analytical placement," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 1711–1714, Mar. 2013.

[6] M. Hsu, V. Balabanov, and Y. Chang, "TSV-aware analytical placement for 3-D IC designs based on a novel weighted-average wirelength model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 32, No. 4, pp. 497–509, Apr. 2013.

[7] W. C. Naylor, R. Donelly, and L. Sha, "Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer," *U.S. Patent 6 301 693*, 2001.

[8] Q. Liu, J. Ma, and Q. Zhang, "Neural network based pre-placement wirelength estimation," *Proceedings of the International Conference on Field-Programmable Technology*, pp. 16–22, Dec. 2012.

[9] J. Chen, J. Kuang, G. Zhao, D. J.-H. Huang, and E. F. Y. Young, "PROS 2.0: A plug-in for routability optimization and routed wirelength estimation using deep learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 42, No. 1, pp. 164–177, Apr. 2022.

[10] D. Hyun, Y. Fan, and Y. Shin, "Accurate wirelength prediction for placement-aware synthesis through machine learning," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 324–327, Mar. 2019.

[11] Z. Xie, R. Liang, X. Xu, J. Hu, C.-C. Chang, J. Pan, and Y. Chen, "Preplacement net length and timing estimation by customized graph neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 41, No. 11, pp. 4667–4680, Nov. 2022.

[12] Z. Wu and I. Savidis, "Transfer of Performance Models Across Analog Circuit Topologies with Graph Neural Networks," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD*, pp. 159–165, Sep. 2022.

[13] Z. Wu and I. Savidis, "Circuit-GNN: A graph neural network for transistor-level modeling of analog circuit hierarchies," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.

[14] Z. Wu, I. Song, and I. Savidis, "Hybrid Utilization of Subgraph Isomorphism and Relational Graph Convolutional Networks for Analog Functional Grouping Annotation," *Proceedings of the IEEE International Symposium on Machine Learning for CAD (MLCAD)*. IEEE, pp. 1–6, 2023.

[15] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. Vandenberg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," *Proceedings of the European Semantic Web Symposium*, pp. 593–607, Jun. 2018.

[16] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1025–1035, Dec. 2017.

[17] C. Albrecht, "IWLS 2005 benchmarks," *In International Workshop for Logic Synthesis : http://www.iwls.org*, Jun. 2005.

[18] P. Shrestha, A. Aversa, S. Phatharodom, and I. Savidis, "EDA-schema: A graph datamodel schema and open dataset for digital design automation," *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 1–8, Jun. 2024.

[19] R. Pascanu, T. Mikolov, and Y. Bengio "On the difficulty of training recurrent neural networks," *Proceedings of the Conference on Machine Learning*, pp. 1310–1318, Jun. 2013.

[20] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, Vol. 7, No. 1, pp. 53040–53065, Apr. 2019.