Transfer Learning of Arrival Time Prediction Models from a 65 nm to a 28 nm Process Node

Pratik Shrestha and Ioannis Savidis
Department of Electrical and Computer Engineering, Drexel University
Philadelphia, Pennslyvania 19104
ps937@drexel.edu, isavidis@coe.drexel.edu

***Abstract*—The ability to adapt predictive models from an older to a newer technology node provides an efficient means to design, optimization, and analyze a circuit. This paper presents a novel application of transfer learning techniques that adapt arrival time prediction models from a 65 nm technology node to a 28 nm technology node. By leveraging models developed for the 65 nm node, the benefits and challenges of transferring the knowledge gained from the 65 nm node to enhance model performance at the 28 nm node are explored. A dataset that includes six IWLS'05 sequential benchmark circuits is utilized to evaluate the effectiveness of the transfer learning between technology nodes. Models trained on 65 nm data are further tuned on 28 nm data and are compared to both models trained on 28 nm data only and 65 nm models utilized directly without further training. The mean absolute error (MAE) and the mean absolute percentage error (MAPE) are calculated for comparison. The proposed transferred models outperform the 65 nm models utilized directly with no further tuning, providing an average improvement of 38.49% in MAE and 28.16% in MAPE, while requiring 1 hour and 40 minutes to generate the additional data needed for modeling tuning. The models trained directly on 28 nm data outperform the proposed transferred models, with an average improvement of 8.51% in MAE and 20.22% in MAPE. However, the improved performance is at a cost of greater computational resources as 2 hours and 47 minutes are needed to generate the additional data and train the 28 nm models. The results underscore the potential of transfer learning to reduce the time and resources needed to adapt predictive models across fabrication technology nodes.**

***Index Terms*—arrival time prediction, physical design, machine learning, graph convolutional networks, transfer learning**

I. Introduction

Machine learning (ML) algorithms are being explored to model various elements and parameters of a circuit [1], [2], with the potential to revolutionize the way the semiconductor industry approaches design optimization and analysis. The application of ML to circuit design ranges from predicting downstream performance metrics such as the timing profile [3], parasitic impedance [4], and power profile [5], to optimizing design stages such as placement [6], clock network synthesis [7], and routing [8]. Adapting machine learning (ML) models to a new technology node is a challenge, particularly due to the large difference between semiconductor manufacturing processes. Transitioning from an older to a more advanced technology node requires modifications to modeling techniques to more accurately predict the behavior of an IC at the advanced node. More precise models are needed to account for improved transistor performance, faster switching speeds, higher heat dissipation per unit area, and quantum mechanical effects [9], [10].

979-8-3503-4953-5/24/$31.00 ©2024 IEEE

Leveraging neural networks with transfer learning [11], [12] provides a means to adapt pre-trained models to a new technology node. The transferred knowledge improves predictions of the behavior of the IC at the more advanced technology node while reducing the amount of data needed to train or fine-tune the models. The utilization of neural networks with transfer learning allows for the efficient use of datasets generated from non-target technology nodes and minimizes the need to train models from the ground up. Consequently, the use of transfer learning significantly reduces the computational time and resources needed to model circuit behavior in a new technology nodes. In this paper, a transfer learning framework is introduced that adapts a graph-based deep neural regression model from a 65 nm technology node to a 28 nm technology node to predict the post-routing arrival time [3] following the completion of the floorplanning of a circuit. A characterization of the performance of the transferred models across technology nodes is performed using six circuits from the IWLS'05 sequential benchmark circuit suite [13].

The paper is organized as follows. Background on transfer learning is provided in Section II. The dataset generated from the 65 nm and 28 nm technology nodes for the modeling and transfer of the prediction of arrival time are described in Section III. A baseline error analysis performed on results provided by a commercial physical design tool is also described in Section III. The proposed framework for the transfer of models that predict arrival time is discussed in Section IV, and an analysis of the results provided by the framework is described in Section V. Some concluding remarks are provided in Section VI.

II. Transfer Learning

Transfer learning leverages knowledge acquired from source domains to enhance performance or prediction in different but related target domains. Assume a domain D consists of a feature space $\mathcal{X}$ and a probability distribution $P(X)$, with $X = \{x_1, ..., x_n\} \in \mathcal{X}$. For a given task T, defined by a label space $\mathcal{Y}$ and a predictive function $f(\cdot)$, the goal of transfer learning is to learn the predictive function $f(\cdot)$ from the domain D that maps input feature vectors to labels in $\mathcal{Y}$. Considering such a problem formulation, model-based transfer learning utilizes common parameters and structures that enable the transfer of knowledge from one trained model to another, which proves effective for applications that leverage deep learning. A source model, $\mathcal{M}_S$, initially trained within the source domain D_S for a source task T_S, is adapted for a target task $\mathcal{T}_T$ in the target domain D_T through the transfer

2024 IEEE 17th Dallas Circuits and Systems Conference (DCAS) | 979-8-3503-4953-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/DCAS61159.2024.10539903

of knowledge to a target model $\mathcal{M}_T$. The key components of model based transfer learning include

- Parameter reuse: Parameters used to train the source model $\mathcal{M}_S$ are utilized to train the target model $\mathcal{M}_T$ for the target task $\mathcal{T}_T$. Parameter reuse is especially beneficial when the source and target tasks include common underlying structures or patterns.
- Architecture reuse: The architecture of the source model is reused and adapted for the target task. Model adaptation includes adding, removing, or modifying layers within the neural network.
- Fine-tuning: The model trained for the source task is further trained (or fine-tuned [14]) on the target task. Fine-tuning typically utilizes a lower learning rate and/or a smaller number of epochs to prevent the loss of generalized features learned from the source task.
- Freezing layers: Select layers of the neural network model are kept static or "frozen" [15] during fine-tuning, which is achieved by not updating the weights of the layers. The technique is used to preserve knowledge learned from current data in the earlier layers of a neural network, which typically learn more general features, while allowing the later layers of a neural network to adapt to the specific data of the target task $\mathcal{T}_T$.

III. Dataset and Baseline Evaluation Analysis

In this work, six IWLS'05 sequential benchmark circuits [13], listed in Table I, are utilized to construct two separate datasets of circuit layouts using both the 65 nm and the 28 nm production ready technology nodes. Synopsys Design Compiler is utilized to synthesize the initial benchmark circuits into a technology-specific gate-level netlist, while Synopsys IC Compiler is utilized to place and route the gate-level netlist. Static timing analysis is performed on the generated circuits using Primetime at each stage of the physical design flow. Interconnect parasitics are extracted by Synopsys IC Compiler and are stored in Standard Parasitic Exchange Format (SPEF) files [16], with data from the files utilized to develop model features. The circuits are synthesized with target design parameters that include an operating frequency of 1 GHz, a circuit aspect ratio of 0.5, and an occupied area of 70% of the total.

TABLE I: IWLS'05 benchmark circuit characteristics.

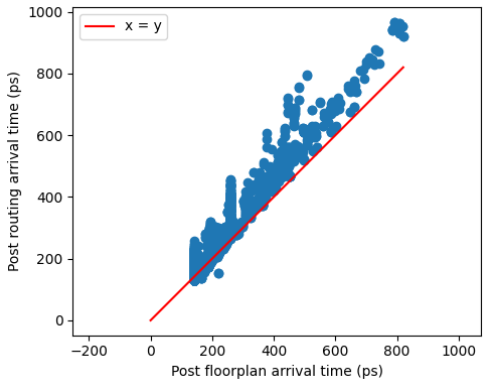
Design	No. of inputs	No. of outputs	No. of Gates		
			Sequential	Combinational	Total
ac97_ctrl	84	48	2199	9656	11855
aes_core	259	129	530	20265	20795
wb_conmax	1130	1416	770	28264	29034
des3_perf	234	64	8808	89533	98341
usb_funct	128	121	1746	11062	12808
pci	162	207	3359	13457	16816

For each technology node the post floorplan arrival time provided by the design tool is considered as the baseline prediction for the post routing timing profile. Scatter plots of the post floorplan and post routing arrival time of the 65 nm dataset and the 28 nm dataset are shown in Fig. 1a and Fig. 1b, repectively. The metrics evaluating the error, specifically the mean absolute percentage error (MAPE) and mean absolute error (MAE), are utilized for analysis and comparison. The MAPE and MAE are given by, respectively,

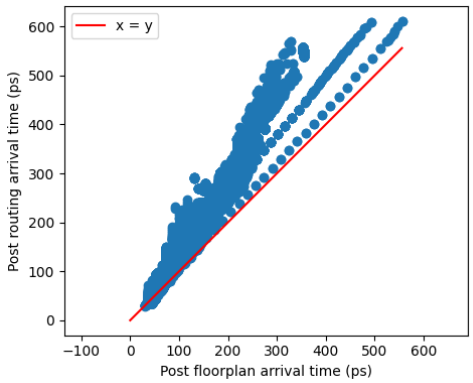
$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}|1-\frac{\hat{y}_i}{y_i}|, \text{ and} \tag{1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|, \tag{2}$$

where n is the total number of timing paths in the dataset, y is the arrival time post routing, $\hat{y}$ is the arrival time post floorplan, and $\bar{y}$ is the average arrival time post routing. The baseline MAE and MAPE errors for each of the circuits are listed in Table III and Table IV, respectively. The scatter plots and the baseline evaluations indicate significant error between the arrival time provided from the two different design stages. The MAE and MAPE for the 28 nm target dataset is, respectively, 20.34% larger and 123.07% larger than that of the 65 nm source dataset.



(a)



(b)

Fig. 1: Post-floorplan and post-routing arrival time estimates provided by PrimeTime for datasets generated from a (a) 65 nm technology node and (b) 28 nm technology node.

IV. Methodology for Arrival Time Prediction

In this section, a transfer learning framework is proposed to adapt a model that predicts the arrival time of the logical paths of a circuit from a 65 nm technology node to a 28 nm technology node. Three key models are trained and evaluated: a source model, a baseline model, and a target transferred model. The source model, trained and evaluated on a dataset

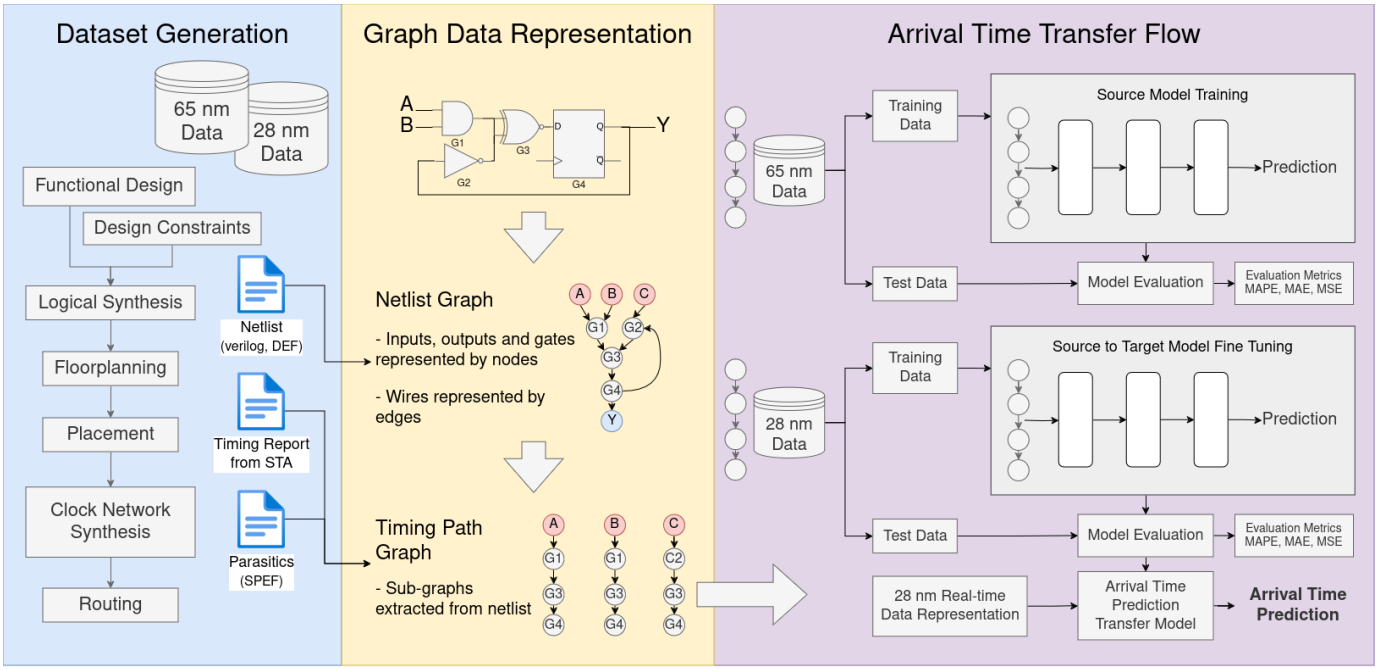


Fig. 2: Dataset generation and transfer learning framework for post floorplan to post routing arrival time prediction across technology nodes.

generated in a 65 nm technology, serves as the foundation for a model fine tuned in a 28 nm technology node. A model that is trained and evaluated on a dataset generated in a 28 nm technology node is utilized as a baseline for comparison, providing a reference against which the transferred model is measured. All models are trained using the same graph structure and utilize identical feature sets and feature engineering methodologies, as detailed in Section IV-A. The source, the baseline, and the fine-tuned target models are described in Section IV-B. The overall execution of the framework and the training flow is shown in Fig. 2.

A. Graph Structure and Feature Engineering

The generated dataset is mapped onto EDA-Schema [17], a graph data model for machine learning applications in design automation. EDA-Schema allows for netlist and timing path graph representations of a circuit extracted from completed stages of the physical design flow. The physical characteristics of the circuit, described in the Verilog and DEF files, are mapped to a netlist graph $G = (V, E)$, where node $v \in V$ represents inputs, outputs, and gates of a netlist and edge $e \in E$ represents a wire connecting two node components. Subgraphs of the timing paths, described as timing path graphs $TPG = (V, E)$, are derived from the netlist graph NG, as shown in Fig. 3b. The timing path subgraphs serve as the primary inputs to a model that predicts the arrival times of gates of a given timing path. Each node of a timing path subgraph is populated with a carefully selected feature set extracted from the netlist and STA timing reports. The feature set of the nodes of the timing paths is listed in Table II.

B. Source, Baseline, and Target Model Architectures

A wide and deep graph neural architecture [3] is used to predict the arrival time. The regression model consists of a graph convolution layer (GCN) [18] connected to six linear layers of size 16. The neural network takes timing path graphs as input. Each hidden layer utilizes a rectified linear unit (ReLU) [19] as an activation function. A gradient clipping

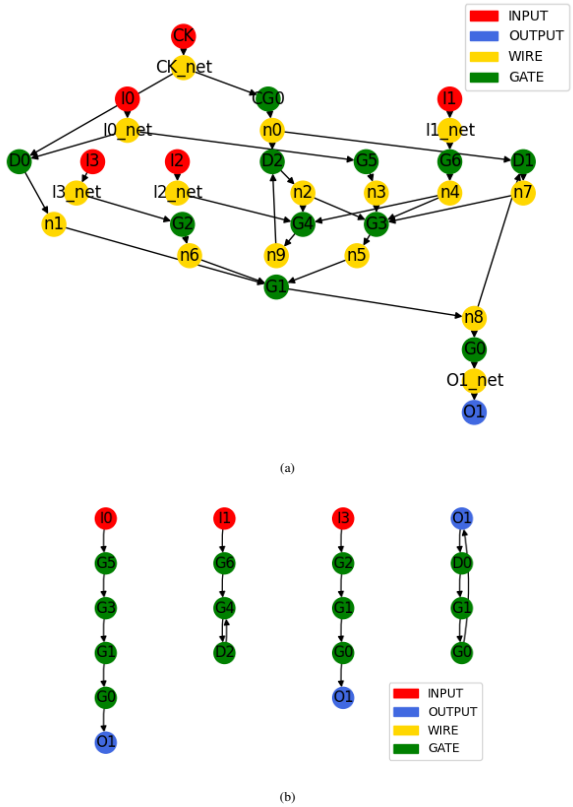


Fig. 3: Benchmark representation of a (a) netlist graph NG and (b) a subset of timing path graphs TPG.

TABLE II: Node features of the netlist and timing path graphs.

Feature Type	Feature	Source
Structural features	Logic level	Calculated from netlist graph representation
	Number of fan-in gates	
	Number of fan-out gates	
	Distance from closest input	
	Distance to closest output	
	Distance from closest flip-flop	
	Distance to closest flip-flop	
Spatial features	Gate co-ordinate (x, y)	DEF file
Current phase timing report features	Gate delay	STA timing report
	Arrival time	
Current phase parasitic features	Total interconnect resistance	SPEF file
	Total interconnect capacitance	

of weights greater than one is applied after each execution of the activation function to avoid exploding gradients [20]. The primary objective function is to minimize the mean squared error (MSE) loss between the actual post routing arrival time y and the predicted estimate of the post routing arrival time $\hat{y}$. The MSE loss is given by

$$MSE_{Loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3}$$

where n is the total number of gates in all of the timing paths of the dataset. Two identical neural networks are separately trained on datasets generated in the 65 nm and 28 nm technology nodes as, respectively, the source and baseline models, which are shown in Fig. 4a and Fig. 4b. The architecture of the source model is modified to implement the architecture of the target model, which is shown in Fig. 4c. The neural network architecture of the target model allows for fine-tuning with transfer learning. The weights of all layers of the source model are frozen to retain the knowledge learned from the 65 nm dataset. An additional layer of size 16 is added beyond the frozen layers of the source model that represents the final layer of the neural network implementing the target model. The weights of the final layer are fined tuned using the 28 nm dataset, which adapts the 65 nm model to the new technology node.

The dataset generated based on the six circuits listed in Table I is utilized for six instances of the source 65 nm deep regression model and six instances of the baseline 28 nm deep regression model. For each instance of the model, one circuit is considered as the test set, and the remaining five circuits are considered as the training set. Each model instance is trained with a stochastic gradient descent (SGD) optimizer [21] for 100 epochs and a batch size of 1024 interconnect graphs for a single iteration of training and validation. An initial learning rate of 0.01 is applied that decays at a rate of 95% every 10 training steps. Early stopping is adopted such that the training process is terminated when there is no improvement in the training loss for 10 epochs, which prevents the model from overfitting to data features by learning the statistical noise in the training dataset. The proposed target deep regression model, which includes an additional layer beyond the frozen 65 nm trained model layers, is fine tuned utilizing the 28 nm dataset for 20 epochs, while utilizing an identical initial learning rate and an identical decay rate as the source model. The framework is implemented in Python 3 using the PyTorch-geometric deep learning library [22]. The training is performed on a system with 96 GB memory, twenty four 4 GHz CPU cores, and an NVIDIA GeForce GTX 3090 graphics card.

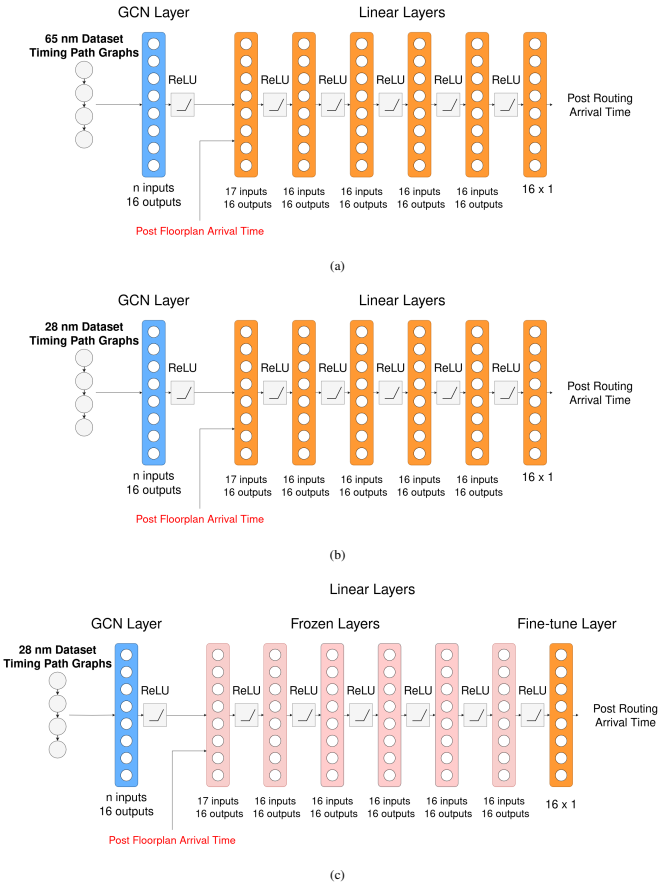


Fig. 4: Graph regression network architecture for the (a) source models, (b) baseline models, and (c) target models.

V. Results of Model Characterization

In this section, an analysis is provided of the results returned by the transfer learning framework for the prediction of arrival time, the methodology of which is described in Section IV. Four scenarios are considered that account for the availability of both data and models from the 65 nm and 28 nm technology nodes. The scenarios include:

- **Preliminary Scenario: Initial 65 nm model**
  The training and prediction on the source technology node is assumed already complete. Therefore, the entire dataset and fully trained models for the 65 nm source technology node are assumed available.
- **Scenario I: 28 nm model**
  A dataset for the 28 nm target technology is generated. A 28 nm baseline model is fully trained using the generated dataset. No transfer learning is attempted.
- **Scenario II: Predict 28 nm arrival times using the 65 nm model**
  The dataset for the 28 nm technology node is not generated. The source 65 nm models are used directly for the prediction of the target 28 nm arrival times without transfer learning.

TABLE III: Analysis of the mean absolute error (MAE) of the models trained on 65 nm data, models trained on 28 nm data, and models transferring learning from the 65 nm data to the 28 nm node.

		ac97_ctrl	aes_core	wb_conmax	des3_perf	usb_funct	pci	average
Preliminary Scenario: Initial 65 nm model	**Tool Prediction**	54.2604	40.6918	62.1978	14.4929	23.0307	22.5641	36.2063
	GCN model (source)	35.9706	17.3582	45.8342	14.8104	20.2539	18.9755	25.5338
Scenario I: 28 nm model	**Tool Prediction**	52.8881	76.5812	32.3576	15.1534	8.5899	75.8574	43.5713
	GCN model (baseline)	14.5319	30.4339	22.7326	9.3403	8.7796	26.5452	18.7273
Scenario II: Predict 28 nm arrival time using 65 nm model		34.9876	57.2522	24.3184	11.5770	5.9399	65.6377	33.2855
Scenario III: Transfer model (65 nm to 28 nm)		30.0791	20.6163	18.4129	31.1933	6.0873	16.4363	20.4709

TABLE IV: Analysis of the mean absolute percentage error (MAPE) of the models trained on 65 nm data, models trained on 28 nm data, and models transferring learning from the 65 nm data to the 28 nm node.

		ac97_ctrl	aes_core	wb_conmax	des3_perf	usb_funct	pci	average
Preliminary Scenario: Initial 65 nm model	**Tool Prediction**	13.94%	11.82%	26.37%	9.39%	9.19%	11.17%	13.65%
	GCN model (source)	8.94%	5.78%	18.41%	9.23%	9.20%	9.23%	10.13%
Scenario I: 28 nm model	**Tool Prediction**	41.96%	41.40%	28.92%	15.38%	12.85%	42.22%	30.45%
	GCN model (baseline)	12.65%	13.36%	14.12%	11.61%	12.70%	12.21%	12.78%
Scenario II: Predict 28 nm arrival time using 65 nm model		26.87%	27.85%	21.36%	13.07%	8.93%	35.74%	22.30%
Scenario III: Transfer model (65 nm to 28 nm)		23.00%	11.36%	12.81%	32.08%	9.65%	7.24%	16.02%

TABLE V: Time required to generate the dataset and to train the models for the source, baseline, and transfer learning scenarios.

		ac97_ctrl	aes_core	wb_conmax	des3_perf	pci	usb_funct	total
65 nm Model	**Dataset Generation**	09m 45s	25m 30s	18m 31s	58m 32s	12m 37s	07m 26s	02h 12m 23s
	Model Training	22m 48s	20m 42s	21m 08s	23m 00s	22m 09s	22m 15s	02h 12m 04s
28 nm Model	**Dataset Generation**	09m 47s	14m 38s	22m 05s	27m 27s	10m 13s	11m 56s	01h 36m 08s
	Model Training	30m 35s	29m 22s	27m 57s	29m 31s	26m 08s	23m 33s	02h 47m 09s
65 nm to 28 nm Transfer Modelling		00m 41s	00m 40s	00m 41s	00m 40s	00m 40s	00m 44s	04m 08s
Preliminary Scenario: Initial 65 nm model (Dataset Generation + Model Training)		32m 34s	46m 13s	39m 39s	01h 21m 32s	34m 46s	29m 42s	04h 24m 27s
Scenario I: 28 nm model (Dataset Generation + Model Training)		40m 22s	44m 01s	50m 03s	56m 59s	36m 22s	35m 30s	04h 23m 18s
Scenario II: requires no new training								
Scenario III: Transfer model (28 nm Dataset Generation + Transfer Training)		10m 28s	15m 18s	22m 46s	28m 08s	10m 54s	12m 41s	01h 40m 17s

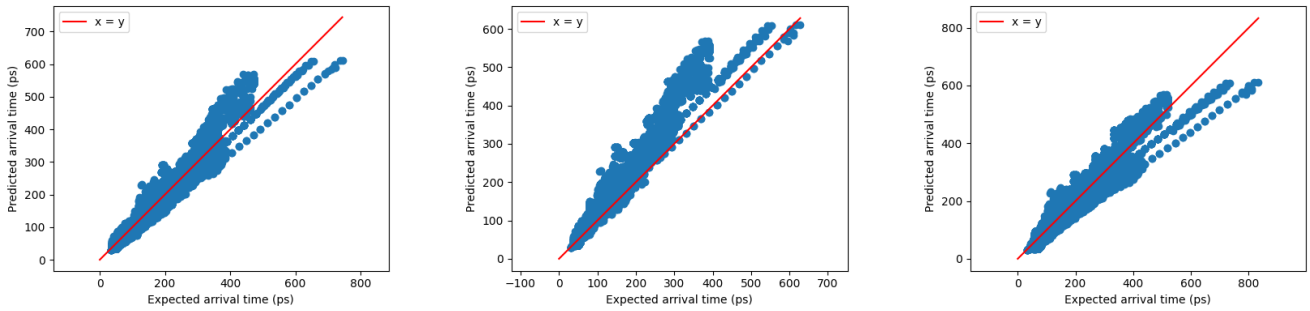


Fig. 5: Expected and predicted arrival time for models considered under (a) Scenario I (28 nm model), (b) Scenario II (predicting 28 nm arrival times using the 65 nm model), and (c) Scenario III (65 nm to 28 nm transfer model).

- **Scenario III: Transfer model (65nm to 28nm)**
  The dataset for the 28 nm technology is generated. The source 65 nm model is fine tuned on the 28 nm dataset, transferring knowledge from the 65 nm node to the target 28 nm node.

The trained regression models are validated against the test datasets. The MAE and MAPE for each prediction scenario are calculated for comparison of the models, with the results listed in Table III and Table IV, respectively. Both the 65 nm and 28 nm models outperform the respective baselines, demonstrating the efficacy of the GCN model architecture. The 28 nm models result in less error as compared to the 65 nm models on average, indicating that the graph neural network provides better predictions for the 28 nm node under identical circumstances. However, when analyzing the graph neural network model of each circuit independently, the 65 nm model outperforms the 28 nm model for certain circuits (*aes_core*, *pci*). Comparing the MAE and MAPE for the three scenarios used to characterize the accuracy of the models that predict the gate arrival time in the 28 nm node, the baseline models considered under Scenario I perform the best on average across all the models as training is completed directly on 28 nm data. The baseline model demonstrates an average of 43.73% lower MAE and 42.69% lower MAPE as compared to the 65 nm source model used directly for 28 nm prediction as described by Scenario II. An outlier prediction of the arrival time is observed for *usb_funct*, where the model considered under Scenario II outperforms the model considered under Scenario I. The target model considered for Scenario III also outperforms the source model utilized for Scenario II with the exception of the *usb_funct* and *des3_perf* circuits, showing an average improvement of 38.49% in MAE and 28.16% in MAPE. The accuracy of the predictions provided by the models is also characterized by comparison of the estimated

and predicted arrival times for Scenarios I, II, and III as shown in Fig. 5a, Fig. 5b, and Fig. 5c, respectively. When analyzing the prediction provided by models for individual circuits, the transfer models considered under Scenario III for *aes_core*, *wb_conmax*, *usb_funct*, and *pci* outperform the respective baseline models considered under Scenario I. From the four circuits that outperform the baseline models, *aes_core* and *pci* were tuned on source GCN models that already performed better than the predicted arrival times provided by the baseline GCN models.

Although the baseline model considered under Scenario I is the most accurate amongst the models trained for each scenario, significantly more time and resources are required for training. A relative comparison of the time required to generate data and train models for each Scenario is also provided, with the results as listed in Table V. The source model considered under Scenario III requires no additional data generation or modeling. For Scenario I, generating a 28 nm dataset and training the models requires 4 hours and 23 minutes. In comparison, for the target transfer models considered under Scenario II, although generation of the dataset requires 1 hour and 25 minutes, the fine tuning of the models only requires an additional 4 minutes. Therefore, the transfer model considered for Scenario III is more accurate than the target models considered for Scenario II and requires less time to train as compared to the baseline models considered under Scenario I.

VI. Conclusions

In this paper, a transfer learning framework that fine tunes a graph-based deep neural regression model is introduced. The framework adapts a source model that predicts the post-routing arrival time after floorplanning of a circuit in a 65 nm technology node to a target model that predicts the post-routing arrival time for the same circuit implemented in a 28 nm technology node. Datasets are generated using a commercial physical design tool for timing path graph representations of six post-floorplan IWLS'05 benchmark circuits, followed by an analysis to establish a baseline error across the six circuits. A baseline transfer learning scenario is considered, where prediction on 28 nm circuits is performed using the 65 nm GCN model directly with no fine-tuning. The transferred models provide better prediction performance as compared to the 65 nm models without the need for further fine-tuning, achieving an average improvement of 38.49% in Mean Absolute Error (MAE) and 28.16% in Mean Absolute Percentage Error (MAPE), while requiring an additional 1 hour and 40 minutes for data generation and model tuning. Models trained specifically on 28 nm data outperform the transferred models, providing an average improvement of 8.51% in MAE and 20.22% in MAPE, but require an additional 2 hours and 47 minutes for data generation and model tuning. The evaluated results highlight the effectiveness of transfer learning in reducing the time and resources needed to adapt models across different semiconductor technology nodes, suggesting a promising strategy towards more efficient and accurate predictive modeling techniques.

References

[1] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, and K. Zhong, "Machine learning for electronic design automation: A survey," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 26, No. 5, pp. 1–46, Jun. 2021.

[2] P. Shrestha and I. Savidis, "EDA-ML: Graph representation learning framework for digital IC design automation," *Proceedings of the International Symposium on Quality Electronic Design (ISQED)*, pp. 241–246, Apr. 2024.

[3] P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 127–133, Sept. 2022.

[4] P. Shrestha and I. Savidis, "Graph representation learning for parasitic impedance prediction of the interconnect," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2023.

[5] Y.-C. Lu, W.-T. Chan, V. Khandelwal, and S. K. Lim, "Driving early physical synthesis exploration through end-of-flow total power prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pp. 97–102, Sept. 2022.

[6] Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via ppa-directed graph clustering," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*. IEEE, pp. 1–6, Sep. 2022.

[7] Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, Nov. 2019.

[8] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," *Proceedings of the IEEE Design, Automation & Test in Europe Conference & Exhibition*, pp. 1–6, Apr. 2007.

[9] E. Ungersboeck, V. Sverdlov, H. Kosina, and S. Selberherr, "Modeling of advanced semiconductor devices," *ECS Transactions*, Vol. 4, No. 1, pp. 207–218, Sep. 2006.

[10] S.K. Springer, S. Lee, N. Lu, E.J. Nowak, J.-O. Plouchart, J.S. Watts, R.Q. Williams, and N. Zamdmer, "Modeling of variation in submicrometer CMOS ULSI technologies," *IEEE Transactions on Electron Devices*, Vol. 53, No. 9, pp. 2168–2178, Oct. 2006.

[11] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, Vol. 109, No. 1, pp. 43–76, 2021 2021.

[12] F. Yu, X. Xiu, and Y. Li, "A survey on deep transfer learning and beyond," *Mathematics*, Vol. 10, No. 19, pp. 3619, Oct. 2022.

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

[14] B. Chu, V. Madhavan, O. Beijbom, J. Hoffman, and Trevor Darrell, "Best practices for fine-tuning visual classifiers to new domains," *Proceedings of the Computer Vision – ECCV 2016 Workshops*. pp. 435–442, Nov. 2016, Springer International Publishing.

[15] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Proceedings of the International Conference on Neural Information Processing Systems*. p. 3320–3328, Dec. 2014, MIT Press.

[16] "IEEE standard for integrated circuit (IC) open library architecture (OLA)," *IEEE Std 1481-2019 (Revision of IEEE Std 1481-2009)*, pp. 1–641, Mar. 2020.

[17] 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.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, pp. 1–14, Sept. 2016.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," *Proceedings of the International Conference on Machine Learning*, pp. 807–814, Jun. 2010.

[20] G. Philipp, D. Song, and J. G. Carbonell, "The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions," *arXiv preprint arXiv:1712.05577*, pp. 1–85, Dec. 2017.

[21] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, pp. 1–14, Sept. 2016.

[22] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," *arXiv preprint arXiv:1903.02428*, pp. 1–9, Mar. 2019.