A Graph Neural Network Method for Fast ECO Leakage Power Optimization

Kai Wang, Peng Cao
National ASIC System Engineering Technology Research Center, Southeast University, Nanjing, China
{seuwangk, caopeng}@seu.edu.com

2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC) | 978-1-6654-2135-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASP-DAC52403.2022.9712486

Abstract— In modern design, engineering change order (ECO) is often utilized to perform power optimization including gate-sizing and Vth-assignments, which is efficient but highly timing consuming. Many graph neural network (GNN) based methods are recently proposed for fast and accurate ECO power optimization by considering neighbors' information. Nonetheless, these works fail to learn high-quality node representations on directed graph since they treat all neighbors uniformly when gathering their information and lack local topology information from neighbors one or two-hop away. In this paper, we introduce a directed GNN based method which learns information from different neighbors respectively and contains rich local topology information, which was validated by the Opencores and IWLS 2005 benchmarks with TSMC 28nm technology. Experimental results show that our approach outperforms prior GNN based methods with at least 7.8% and 7.6% prediction accuracy improvement for seen and unseen designs respectively as well as 8.3% to 29.0% leakage optimization improvement. Compared with commercial EDA tool PrimeTime, the proposed framework achieves similar power optimization results with up to 12X runtime improvement.

Keywords—GNN, ECO, directed graph, V_{th} assignments

I. Introduction

In modern design, leakage power always takes up a significant portion of total power, especially in advance process corners or in high operating temperature [1], which causes power wastage and leads to a potential thermal runaway. Therefore, Engineering Change Orders (ECOs) are often utilized to optimize power, performance and area(PPA) to achieve best-possible design PPA, which always takes multiple iterations to lead up to tapeout due to the complexity of leakage optimization.

At signoff ECO stage, gate-sizing and V_{th}-assignment are always utilized to optimize leakage as well as circuit timing since each standard cell today includes multiple types with different V_{th} and different gate lengths. V_{th}-assignment is the preferable leakage optimization approach in signoff ECO stage as it not only optimizes the leakage power but also not creates significant disturbance to the overall placed and routed layout. It is very effective, but requires much longer runtime due to iterative nature of cell swap and subsequent timing check with correction. In addition, circuit timing is tighter in ECO stage, which further contributes to its longer runtime [2].

This leakage optimization problem has been proven to be NP-hard [3]. For example, consider a design with N cells and a standard cell library with 3 V_{th} choices. If we just consider V_{th} swaping for power optimization, the search space is 3^N. Therefore, it is almost impossible for designers to effectively leverage all the optimization choices, which makes the runtime of ECO leakage optimization a real concern. Hence, learning-based methods are increasingly explored to solve this problem.

There have been some learning-based methods [4], [5] utilized in leakage optimization problem. But these methods do not achieve satisfying results, since the final V_{th} assignment of given cell is not only determined by the information of the cell itself, but highly depended on its neighbors. Therefore, it is not enough to only use cell's own information to predict the final V_{th} type, but also necessary to take the neighboring node's information into account. Graph neural networks(GNNs) are considered as a promising graph learning based method to solve this problem. The common GNN models include graph convolution network(GCN) [6], graph attention network(GAT) [7] and GraphSAGE(GSAGE) [8]. These models can learn the given node's updated representation by convolving its own representation with its neighbors' representations and thereby topology pattern is learned.

However, there are still two drawbacks for common GNN models to be utilized in leakage optimization problem, which limits their ability to achieve high-quality V_{th}-assignment results compared with Synopsys PrimeTime. First, for a given cell in the netlist, V_{th} swap of different neighboring cells causes different impact on the final V_{th} assignment itself, which means our model should have the ability to learn information from different neighbors including fanins, fanouts and siblings respectively [9]. Second, common GNN models with multiple convolutional layers only preserve the embedding of final layers when learning node representations, which loses much local topology information from neighboring cells one or two-hop away [10].

In this work, we propose DGLPO, a directed graph neural network method for fast ECO leakage power optimization. Different from common GNN models which treat all neighbors uniformly when gather their information and lack local topology information, our model is able to learn high-quality node representation on directed graph by aggregating the information of different neighboring nodes including fanins, fanouts and siblings respectively as well as concatenating embeddings from all layers as final node representations. Compared with common GNN models, our proposed model achieves more accurate V_{th}-assignment predictions in both seen designs and unseen designs. Besides, our model achieves comparable optimization results with up to 12X runtime improvement compared with Synopsys PrimeTime.

The rest of this paper is organized as follows: Section II provides related work on leakage power optimization and estimation. We give details of our modeling flow and methodology in Section III. Section IV reports our experimental design, setup and results. Followed by

978-1-6654-2135-5/22/$31.00 ©2022 IEEE

conclusions in Section V.

II. Related Works

Power optimization research has been widely studied in recent years. Past work can be divided into the following categories:

Non-analytic methods: This catagory mainly includes sentivity guided heuristics [11], [12], dynamic programming [13], [14], network flow [15], [16] and other methods to find feasible solutions. However, these methods are often highly time-consuming due to iterative nature of swap and timing check with correction. Moreover, they are also not transferable and therefore cannot be generalized to unseen designs.

Analytic methods: This kind of methods perform leakage power optimization by linear programing [17], [18] or Lagrangian optimization problem [19], [20] to solve the problem. The goal is to reduce the leakage power consumption as much as possible and not bring new timing violations. Nonetheless, for large and complex designs, these methods usually take tens of hours of running time, which makes it difficult to be used in real-world usage.

Learning-based methods: In recent years, some machine learning methods have been proposed for leakage power optimization problems. The work of [4] proposes a regression model to predict the changes in timing slack brought about by V_{th}-swap. The work of [5] uses the support vector machine(SVM) to predict the power and timing optimization results to improve the quality of the optimization and reduce the time consumption. However, these models ignore the huge influence of neighboring cells on the final optimization results. Therefore, these methods can not perform power optimization accurately without considering the feature of neighbors.

GNN based methods: At present, there have been some GNN based methods for leakage optimization or leakage reduction estimation. GNN models are graph version of convolutional neural network(CNN) as they also contains of several sequential convolution layers. They all convolve each node's representation with its neighbors' representations to derive an updated representation for the given node, thereby topology pattern is learned. The work of [21] uses GraphSAGE model to predict the final V_{th} assignments , and analyzes the explanation of the GraphSAGE model. In [22], the GCN model is used to predict the V_{th} assignments, and a heuristic algorithm is used to repair the remaining timing violations and minimum implant width(MIW) violations. The work of [23] uses GAT model to predict the leakage reduction, which is very close to the result of commercial tool optimization. However, these models are unable to distinguish the impact from different neighbors, which limits its power to learn high-quality node embedding on directed graphs. Hence, the final prediction accuracy can be further improved.

III. Approach

A. Overview

The proposed DGLPO model is illustrated in Fig.1. For a given netlist shown in Fig.1(a), it includes central node d, fanins(instances$\{a,\ b\}$), fanouts(instances$\{f,\ g,\ h\}$) and siblings(instances$\{c,\ e\}$). Then we convert it into a directed acyclic graph(DAG) and construct an adjacency

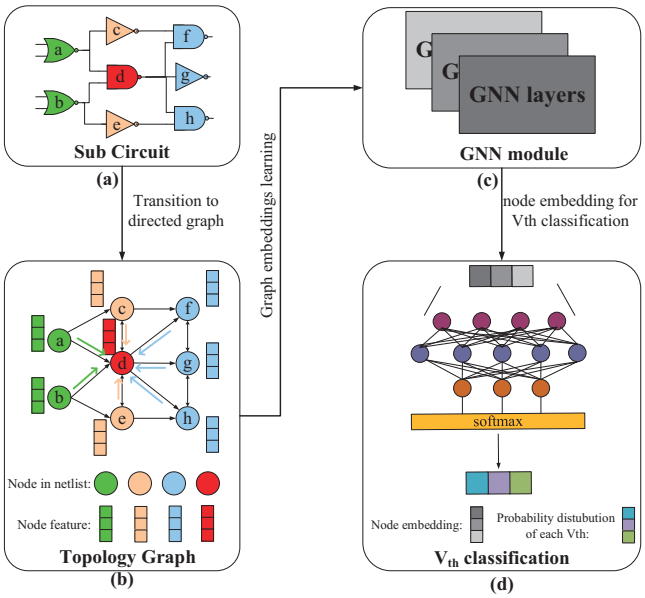

Fig. 1. Overall flow - GNN based learning of V_{th} assignments with consideration of different neighboring cells and local topology information, (a)input netlist, (b)graph embedding learning, (c)GNN module, (d)V_{th} classification

matrix from this netlist. Meanwhile, we extract features of each node as feature matrix and final V_{th}-assignments as ground-truths in Fig.1(b). It is worth noting that information from different neighbors is taken into different consideration, when aggregating their representations to derive an updated representation for the given node d. Then, this topology graph is inputted into our GNN module for graph embeddings learning, which contains multiple layers as shown in Fig.1(c). In this part, we concatenate the outputs from all layers as final node embedding with more local topology information. Both of these two operations enable our model to learn higher-quality node representations. Finally, as illustrated in Fig.1(d), after our GNN model learns an embedding for each node, the embeddings of the nodes are inputted into the downstream classification network, and the accurate V_{th}-assignment prediction for each node is generated by our framework.

B. Adjacency Matrix Construction

In order to use our model for predicting V_{th} assignment results, we construct an adjacency matrix for each given netlist. Each netlist is regarded as a DAG, where each cell in netlists is regarded as a node, and the connection between these cells is represented by an edge.

Then we define an empty $N \times N$ matrix A while N denotes the number of cells in a given netlist. Then, each node of DAG is assigned with a number between 1 to N without overlap. when a node i is connected to a node j, an element on i-th row and j-th of matrix A is set to 1. Furthermore, V_{th}-assignments of the different neighboring nodes bring different effect to whether swapped cell is fanin, fanout or sibling of the given cell. For a given netlist in Fig.1(a), we construct an adjacency matrix with consideration of topology connectivity information as shown in Fig.2. In this way, for each graph convolutional layer we use 3 different weight matrices for

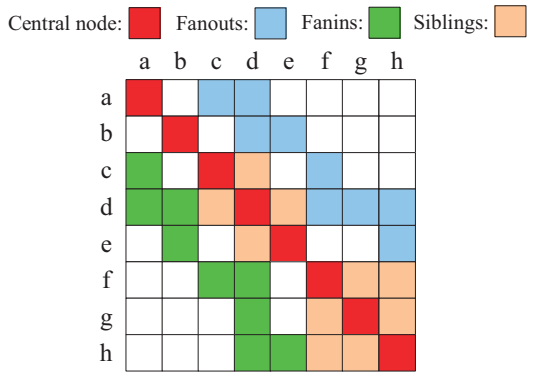

Fig. 2. Adjacency matrix construction with consideration of topology information

fanins, fanouts and siblings, which enables our model to learn different information from cells by its connection relations since those weights are calculated independently. Finally, the adjacency matrix A is translated into compressed sparse row(CSR) format [24] due to its large sparsity, which makes the space complexity pseudo-linear instead of quadratic with respect to the netlist size.

C. Feature Selection

TABLE I
INITIAL NODE FEATURES FOR FEATURE MATRIX CONSTRUCTION

Parameter	Description
Features about the given node	
wst output slack	worst slack of output pin
wst input slack	worst slack of input pin(s)
max output slew	max transition time of output pin
max input slew	max transition time of input pin(s)
tot input cap	sum of input pin(s) capacitance
ini leakage power	node's initial leakage power
max delay change	max delay change from all swaps
propagation delay	propagation time of the node
Feature about its neighboring cells	
wst fanin slack	worst slack of fanins
tot fanin cap	total capacitance of fanins
wst fanout slack	worst slack of fanouts
tot fanout cap	total capacitance of fanouts
wst sibling slack	worst slack of siblings
total sibling cap	total capacitance of siblings

In order to enable our model to perform tool-accurate signoff power optimization without introducing new timing violations, we extract the following features from timing reports and SPEF files. Parameters used to construct the feature matrix of GNN models are shown in Table I. Most of these features are related to timing or power of given design. As mentioned above, in the process of leakage power optimization, the final V_{th} assignment of each node not only depends on its own features, but also highly depends on the features of its neighbors. Therefore, we extract node's own features with respect to slack, transition time, capacitance as well as propagation delay and so on. For example, a cell with large slack margin and small delay increase has higher priority to be swapped. For features related to transition time, an increase in trainsition time leads to the delay of fanouts increase, which affects the swapping possibility of neighboring cells. Moreover, these information of its neighbors including fanins, fanouts and siblings is also taken into account. These features are chosen according to the domain knowledge and parameter sweeping experiments in this work.

Algorithm 1 GNN embedding generation algorithm

Input: (1)DAG: netlist graph, (2)$A_{N\times N}$: adjacency matrix, (3)h_v^0, $\forall v \in V$: initial feature matrix, (4)Y: ground-truths, (5)K: depth, (6)$W^k, \forall k \in \{1, 2, ..., K\}$: weight matrix at k^{th} layer, (7)AGG_{in}, AGG_{out}, AGG_{sib}: aggregator functions used to aggregate information from fanins, fanouts and siblings, (8)α: learning rate (9)$\{\beta_1, \beta_2\}$: Adam parameters.
Output: node representation h_v^K for all $v \in V$

0: $h_v^0 = \frac{h_v^0}{\|h_v^0\|_2}, \forall v \in V$
0: **for** $k = 1, ..., K$ **do**
0: **for** $\forall v \in V$ **do**
0: $h_{fanin}^k = \mathrm{AGG_{in}}\left(\left\{h_u^{k-1}, \forall u \in N_{in}^{k-1}(v)\right\}\right)$
0: $h_{fanout}^k = \mathrm{AGG_{out}}\left(\left\{h_u^{k-1}, \forall u \in N_{out}^{k-1}(v)\right\}\right)$
0: $h_{sib}^k = \mathrm{AGG_{sib}}\left(\left\{h_u^{k-1}, \forall u \in N_{sib}^{k-1}(v)\right\}\right)$
0: $h_v^k = MLP^k\left(h_{fanin}^k \| h_{fanout}^k \| h_{sib}^k \| h_v^{k-1}\right)$
0: **end for**
0: $h_v^k = \frac{h_v^k}{\|h_v^k\|_2}, \forall v \in V$
0: **end for**
0: $h_v^K = \left(h_v^1 \| h_v^2 \| ... \| h_v^K\right)$
0: **for** $\forall v \in V$ **do**
0: $p_v \leftarrow \mathrm{softmax}\left(\mathbf{W}_k^{NN} \cdot h_v^K\right)$
0: $g_v \leftarrow \nabla_\theta \left[\sum_{c=1}^{n} Y_{ic} \log\left(p_{vc}\right)\right]$
0: $\{\mathbf{W}_k\} \leftarrow \mathrm{Adam}\left(\alpha, \{\mathbf{W}_k\}, g_v, \beta_1, \beta_2\right)$
0: **end for**
=0

D. DGLPO Model

Our proposed model is illustrated in Algorithm I. First, normalization of node representations h_v^k is utilized in Line 1 and Line 9 for better convergence. Then, in Lines 2-8, different from common GNN models mentioned above, it is worth noting that we use AGG_{in} to aggregate the information of fanins per node $N_{in}^{k-1}(v)$ to generate the aggregated embedding vector h_{fanin}^k of the fanin set. Then our model generates the aggregated embedding vector h_{fanout}^k and h_{sib}^k of the fanout set and sibling set in a similar way respectively. These three aggregated embeddings including h_{fanin}^k, h_{fanout}^k and h_{sib}^k are concatenated with previous node embedding h_v^{k-1} and then fed into a multilayer perceptron(MLP) module to update the given node embedding h_v^k.

Also of note, inspired by many classical deep learning methods in Euclidian space which combines shallow and deep layers [25], we concatenate the outputs from all

layers in the final embedding h_v^K in Line 11, rather than just retaining the embedding of last layer like other GNN model does. This is because each embedding of the layers includes information from different depths, The shallower one includes more local information, while deeper one contains more global information. By this way, the final node representation contains more information from its neighbors, which enables classification network to make more accurate predictions and lead to better convergence. In this paper, we implement AGG_{in}, AGG_{out} and AGG_{sib} with the MEAN aggregator, which is proved more effective than other aggregation functions such as POOLING method and LSTM method in our work. Finally, we calculate the cross-entropy loss from V_{th} classification network, and adaptive moment estimation(Adam) [26] is leveraged to update the parameters in the framework by minimizing the loss function in Line 12-16.

IV. Experimental Results And Discussion

A. Experimental Setup

In this section, we took a set of citcuits from Opencores.org and IWLS 2005 [27] benchmarks for training and testing. All these designs were synthesized with TSMC 28nm standard cell library by Synopsys Design Compiler, where each cell had one of three V_{th}-types including ULVT, LVT and RVT, and placed and routed using Synopsys IC Compiler. Lastly, we used Synopsys PrimeTime to perform commercial timing analysis and ECO leakage power optimization to get the initial features and final V_{th} assignments as ground-truths. Table II illustrated the characteristics of these designs, In these designs, there were 7 designs utilized in the training process, and other 5 designs were used as unseen circuits to validate the usefulness of our model in design-specific and design-independent usecase respectively. For fair comparison, we followed the experimental settings of [21], where all the cells were initially in the lowest V_{th} type(tightest timing constraint). In this work, all GNN models were implemented in Python3 with Pytorch 1.4 and Pytorch-geometric. The experiment was performed on a Linux machine with NVIDIA Tesla V100 GPU. The total memory used in the training is 32GB.

For hyper-parameters of GNN models, there were three layers for each GNN model, with the hidden dimension of 128. We used a learning rate α of 0.001, a weight decay of 0.0001 and chosen ReLU as activation function for all models. The Adam parameters β_1 and β_2 were 0.9 and 0.999 respectively. A batch normalization layer was applied after each layer for accelerating the overall training process and better convergence. In order to evaluating V_{th} classification accuracy, F1-score was chosen as our main metric for comparison, which represented a balance between precision and recall, it took values from 0 to 1, while values closer to 1 mean higher accuracy.

In order to validate that our model is more effective for learning V_{th}-assignments than other GNN models, we compare our model with GCN [22], GAT [23] and GraphSAGE [21] in seen designs and unseen designs respectively. For fair comparison, all of them were trained and tested by the identical training data and testing data and use same features shown in Table II.

TABLE II
Our benchmarks and their attributes

Benchmarks	# FFs	# Cells	Category
des_perf	8648	51064	Train
pci_bridge	3582	7420	
mem_ctrl	1126	5826	
ethernet	10544	42618	
aes	670	11307	
gfx	1477	10654	
wb_dma	718	41156	
nova	29805	138990	Test
vga_lcd	17071	58816	
tate	31420	189477	
ecg	7680	51846	
wb_conmax	818	36333	

B. Comparison of GNN Models

For seen designs, we train the models for each training design, and test the models on same designs which are synthesized, placed and routed under different clock period and parameters. Fig.3(a) illustrates the results of our model and others for seen designs. All GNN models above achieve good prediction accuracy benefited from the ability of aggregating information of the neighboring nodes. Moreover, our model shows satisfying accuracy with the F1-score range from 92.1% to 96.4%. Compared to other GNN models, our model achieves average accuracy improvements up to 7.8% at least, which proves that learning the information from different neighboring nodes leads to better accuracy and convergence.

For unseen designs, we train the models on the training circuits and test the models on the test circuits which were not trained. Fig.3(b) shows the results of our model and other GNN models. From the figure, we can see that our model still achieves the best performance than other GNN models with the F1-score range from 91.3% to 94.5%, and improves them at least 7.6% on average. Our insight is that our model can learn the different influence of fanins, fanouts and siblings respectively in leakage optimization and generate high-quality node representations by preserve all embeddings of different layers, while other models fail to distinguish the information of different neighboring nodes and only use the output of the final layer.

Furthermore, we analyze the embeddings for all cells learned by our model by ploting the tSNE [28] plot of vga_lcd design's embeddings in Fig.4. We can see that clustered points are comprised of same V_{th} type, which demonstrates that our model has the ability to learn high-quality node embeddings, therefore, our model can always perform high-quality V_{th}-assignments.

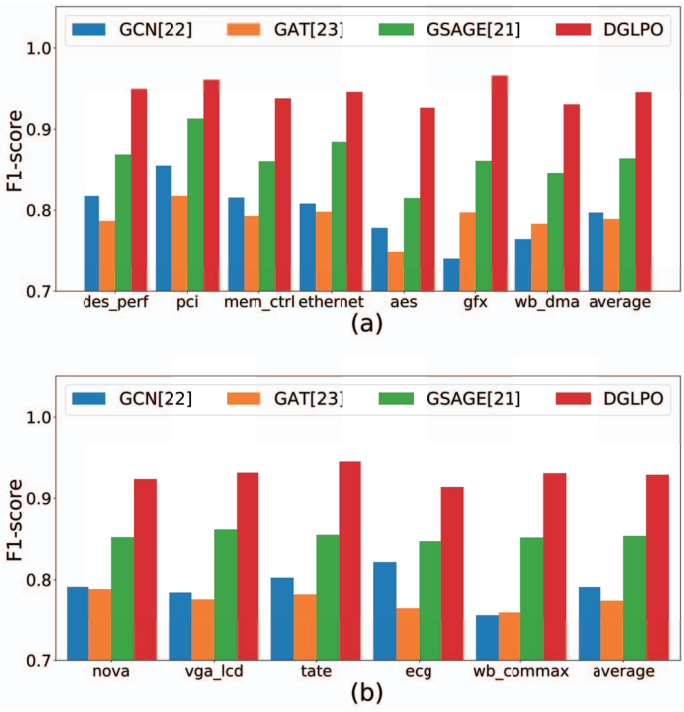

Fig. 3. F1-score between prediction and label comparison with competitive models in (a) seen designs and (b) unseen designs

Finally, the optimization results on leakage and timing for unseen designs are shown in Table III. It is worth noting that the WNS of each unseen circuit before optimization is close to 0 for reasonable signoff power optimization, and the optimization constraints are that the WNS and TNS do not degrade and no new violation is introduced. Compared with other models, we can see that our model achieves leakage improvement range from 8.3% to 29.0% at least for unseen designs with similar runtime. Moreover, our model achieves similar optimization quality with runtime improvement by up to 12X compared with Synopsys PrimeTime.

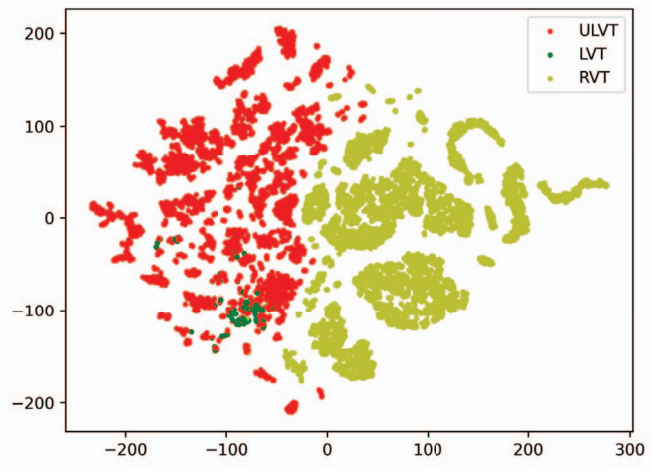

Fig. 4. tSNE result of 128 dim embeddings for all cells in vga_lcd design, color based on ground-truth V_{th}-assignment per cell by PrimeTime

V. Conclusions

We present DGLPO based on graph learning to predict final V_{th} assignments with consideration of different effect of different neighbors and local topology information. Experimental results demonstrate that the proposed framework achieves significant prediction precision improvement compared with common GNN models and provides commercial-quality signoff power optimization results on unseen designs instantly.

Acknowledgement

This work was supported in part by the National Key Research and Development Program of China (Grant No. 2019YFB2205004), and in part by the National Natural Science Foundation of China under Grant(62174031) and in part by the Jiangsu Natural Science Foundation (Grant No. BK20201233).

References

[1] Ahmed T El-Thakeb, Hamdy Abd Elhamid, Hassan Mostafa, and Yehea Ismail. Performance evaluation of finfet based sram under statistical vt variability. In *2014 26th International Conference on Microelectronics (ICM)*, pages 88–91. IEEE, 2014.

[2] Tao Luo, David Newmark, and David Z Pan. Total power optimization combining placement, sizing and multi-vt through slack distribution management. In *2008 Asia and South Pacific Design Automation Conference*, pages 352–357. IEEE, 2008.

[3] Wing Ning. Strongly np-hard discrete gate-sizing problems. *IEEE transactions on computer-aided design of integrated circuits and systems*, 13(8):1045–1051, 1994.

[4] Shuhan Bao. Optimizing leakage power using machine learning. *CS229 Final Project*, 2010.

[5] SLPSK Patanjali, Milan Patnaik, Seetal Potluri, and V Kamakoti. Mltimer: Leakage power minimization in digital circuits using machine learning and adaptive lazy timing analysis. *Journal of Low Power Electronics*, 14(2):285–301, 2018.

[6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[8] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[9] Ecenur Ustun, Chenhui Deng, Debjit Pal, Zhijing Li, and Zhiru Zhang. Accurate operation delay prediction for fpga hls using graph neural networks. In *Proceedings of the 39th International Conference on Computer-Aided Design*, ICCAD '20, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Zhiyao Xie, Rongjian Liang, Xiaoqing Xu, Jiang Hu, Yixiao Duan, and Yiran Chen. Net2: A graph attention network method customized for pre-placement net length estimation. *CoRR*, abs/2011.13522, 2020.

[11] Jin Hu, Andrew B Kahng, SeokHyeong Kang, Myung-Chul Kim, and Igor L Markov. Sensitivity-guided metaheuristics for accurate discrete gate sizing. In *Proceedings of the International Conference on Computer-Aided Design*, pages 233–239, 2012.

[12] Santiago Mok, John Lee, and Puneet Gupta. Discrete sizing for leakage power optimization in physical design: A comparative study. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 18(1):1–11, 2013.

[13] Muhammet Mustafa Ozdal, Steven Burns, and Jiang Hu. Gate sizing and device technology selection algorithms for

TABLE III
RESULTS ON LEAKAGE, TIMING AND RUNTIME BETWEEN GAT [23], GCN [22], GSAGE [21], DGLPO AND PRIMETIME. ALL DESIGNS BEFORE ECO OPTIMIZATION ARE USING ULTRA-LOW V_{th} FOLLOWING SETTINGS OF [21]. NOTE THAT THESE DESIGNS ARE UNSEEN IN TRAINING PROCESS FOR ALL GNN MODELS

Benchmark	Optimization Engine	Leakage Power (mW)	WNS (ps)	TNS (ps)	Runtime (s)
nova	Pre-recovery	14.8	-10.2	-42.3	-
	PrimeTime	2.17	-4.11	-12.4	110
	GAT [23]	2.57	-4.95	-24.9	12.3
	GCN [22]	2.58	-8.86	-24.8	10.5
	GSAGE [21]	2.54	-7.82	-35.8	11.1
	DGLPO	2.33(8.3%)	-4.83	-20.7	11.9(9X)
vga_lcd	Pre-recovery	16.1	-10.3	-129.4	-
	PrimeTime	1.16	-8.32	-82.3	68
	GAT [23]	1.83	-8.44	-93.4	6.6
	GCN [22]	1.69	-8.53	-93.8	5.8
	GSAGE [21]	1.58	-9.34	-106.7	5.7
	DGLPO	1.37(13.3%)	-8.46	-93.1	6.1(11X)
tate	Pre-recovery	41.1	-14.7	-90.1	-
	PrimeTime	2.61	-13.0	-17.5	167
	GAT [23]	4.57	-14.5	-18.7	17.1
	GCN [22]	4.20	-14.5	-20.6	16.9
	GSAGE [21]	3.41	-14.3	-18.9	15.5
	DGLPO	2.81(17.6%)	-14.0	-18.1	16.8(10X)
ecg	Pre-recovery	12.3	-17.1	-558.1	-
	PrimeTime	0.83	-0.11	-1.5	143
	GAT [23]	1.09	-1.22	-4.5	14.9
	GCN [22]	1.24	-2.22	-19.8	13.8
	GSAGE [21]	1.08	-2.88	-35.3	13.2
	DGLPO	0.85(21.3%)	-1.12	-3.9	14.1(10X)
wb_conmax	Pre-recovery	3.57	-27.1	-39.6	-
	PrimeTime	0.21	-20.1	-25.8	63
	GAT [23]	0.34	-21.2	-27.0	5.6
	GCN [22]	0.39	-21.2	-27.0	5.3
	GSAGE [21]	0.31	-21.2	-27.4	5.1
	DGLPO	0.22(29.0%)	-20.9	-25.9	5.4(12X)

high-performance industrial designs. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 724–731. IEEE, 2011.

[14] Yifang Liu and Jiang Hu. A new algorithm for simultaneous gate sizing and threshold voltage assignment. *IEEE Transactions on Computer-aided design of integrated circuits and systems*, 29(2):223–234, 2010.

[15] Li Li, Peng Kang, Yinghai Lu, and Hai Zhou. An efficient algorithm for library-based cell-type selection in high-performance low-power designs. In *Proceedings of the International Conference on Computer-Aided Design*, pages 226–232, 2012.

[16] Huan Ren and Shantanu Dutt. A network-flow based cell sizing algorithm. *structure*, 1:N3, 2008.

[17] David G Chinnery and Kurt Keutzer. Linear programming for sizing, vth and vdd assignment. In *Proceedings of the 2005 international symposium on Low power electronics and design*, pages 149–154, 2005.

[18] Kwangok Jeong, Andrew B Kahng, and Hailong Yao. Revisiting the linear programming framework for leakage power vs. performance optimization. In *2009 10th International Symposium on Quality Electronic Design*, pages 127–134. IEEE, 2009.

[19] Vinicius S Livramento, Chrystian Guth, Jose Luis Guentzel, and Marcelo O Johann. A hybrid technique for discrete gate sizing based on lagrangian relaxation. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 19(4):1–25, 2014.

[20] Ankur Sharma, David Chinnery, Sarvesh Bhardwaj, and Chris Chu. Fast lagrangian relaxation based gate sizing using multi-threading. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 426–433. IEEE, 2015.

[21] Yi-Chen Lu, Siddhartha Nath, Sai Surya Kiran Pentapati, and Sung Kyu Lim. A fast learning-driven signoff power optimization framework. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2020.

[22] Wonjae Lee, Yonghwi Kwon, and Youngsoo Shin. Fast eco leakage optimization using graph convolutional network. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, pages 187–192, 2020.

[23] Uday Mallappa and Chung-Kuan Cheng. Gra-lpo: Graph convolution based leakage power optimization. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 697–702. IEEE, 2021.

[24] James B White and Ponnuswamy Sadayappan. On improving the performance of sparse matrix-vector multiplication. In *Proceedings Fourth International Conference on High-Performance Computing*, pages 66–71. IEEE, 1997.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Christoph Albrecht. Iwls 2005 benchmarks. In *International Workshop for Logic Synthesis (IWLS): http://www. iwls. org*, 2005.

[28] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.