A Fast Learning-Driven Signoff Power Optimization Framework

Yi-Chen Lu
yclu@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Siddhartha Nath
nath@synopsys.com
Synopsys Inc.
Mountain View, California

Sai Surya Kiran Pentapati
sai.pentapati@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Sung Kyu Lim
limsk@ece.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

ABSTRACT

Modern high-performance System-on-Chip (SoC) design flows highly depend on signoff tools to perform timing-constrained power optimization through Engineering Change Orders (ECOs), which involve gate-sizing and V_{th}-assignment of standard cells. However, ECOs are highly time-consuming, and the power improvement is unknown in advance. Ever since the industrial benchmarks released by the ISPD-2012 gate-sizing contest, active research has been conducted extensively. Nonetheless, previous works were mostly based on heuristics or analytical methods whose timing models were oversimplified and lacked formal validations from commercial signoff tools. In this paper, we propose ECO-GNN, a transferable graph-learning-based framework, which harnesses graph neural networks (GNNs) to perform commercial-quality signoff power optimization through discrete V_{th}-assignment. Our framework generates tool-accurate optimization results *instantly* on unseen netlists that are not utilized in the training process. Furthermore, we implement a GNN-based explanation method to interpret the optimization results achieved by our framework. Experimental results on 14 industrial designs, including a RISC-V-based multi-core system and the renowned ISPD-2012 benchmarks, demonstrate that our framework achieves up to 14X runtime improvement with similar signoff power optimization quality compared with *Synopsys PrimeTime*.

1 INTRODUCTION

Engineering Change Orders (ECOs) are regularly used in modern physical design (PD) flows to optimize power, performance and area (PPA). Every top semiconductor design company runs multiple iterations of signoff ECO to achieve the target PPA. However, in advanced technology nodes, power optimization has become much more complicated than optimizations on other design metrics such as wirelength and timing, which is mainly due to the dominance of leakage power and its complicated relation with the dynamic power [14]. Even though design implementation tools have developed various power optimization techniques throughout the years, designers still heavily rely on signoff tools to recover power at the signoff stage using ECO change-lists that involve gate-sizing and V_{th}-assignment.

Since gate-sizing requires further legalization and routing to validate the design after the optimization, V_{th}-assignment is the preferred approach during signoff ECO as it causes minimum disturbance to the overall placed and routed layout. In *Synopsys PrimeTime*, V_{th}-assignments during signoff ECO not only optimize the leakage power, but also reduce the dynamic power simultaneously [21]. Nonetheless, this optimization conducted by *PrimeTime* is time-consuming and the tool itself remains a blackbox for designers. Therefore, in this work, we aim to develop a fast, explainable signoff power optimization framework that has the ability to perform commercial quality signoff power optimization instantly as well as the facility to explain the achieved optimization results.

V_{th}-assignment refers to assigning an appropriate V_{th} type for each design instance from a set of standard cell libraries to perform power optimization without violating timing constraints [10]. Note that for a given design instance, all the available V_{th} types have the same footprint, and the total number of the available types is limited to the discrete values of threshold voltages specified by the technology. This optimization problem is proven to be NP-hard [11], which implies great opportunities to employ machine learning techniques for solving this problem.

Modern commercial signoff tools perform signoff power ECO based on sophisticated in-house timing models. The models precisely calculate the timing budget for every design instance to help the signoff engines conduct timing-constrained power optimization. The optimization results achieved by these tools are considered as golden QoR in the industry, however, there are two significant drawbacks in the current industrial signoff flows, namely:

- **Extremely long runtime.** A signoff power ECO run often takes several days on an industrial scale design and requires human-in-the-loop for enhancement, which drastically bottlenecks the chip development process.
- **Obscure improvement.** The power improvement is unknown in advance. Designers tend to run multiple optimization configurations in parallel in order to select the best one in the end, which consumes significant amount of computing resources.

In this work, we overcome the above issues by presenting ECO-GNN, which is a graph-learning-based framework that leverages

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415711

graph neural networks (GNNs) to perform V_{th}-assignments for fast signoff power optimization. After learning supervisedly on several designs with the assignment ground-truths given by *Synopsys PrimeTime*, our framework has the ability to perform tool-accurate signoff power optimization on *unseen* designs instantly without degrading the performance or introducing new design rule violations (DRVs). To validate our framework, we consider *Synopsys PrimeTime* as our baseline, and demonstrate that ECO-GNN achieves comparable optimization results with up to 14X runtime improvement on the ISPD-2012 benchmarks [12] and other real-world designs, including a RISC-V based multi-core system.

The goal of this work is to provide designers a fast and accurate signoff power optimization framework with high fidelity as the industry-standard commercial tool, *Synopsys PrimeTime*. The key contributions of this paper are summarized as follows:

(1) Our first major finding is that ECO-GNN learns the behaviour of *Synopsys PrimeTime* effectively and generates comparable optimization results at inference time.
(2) Our second major finding is that ECO-GNN generally shows better power saving but worse timing saving compared with *Synopsys PrimeTime*. This indicates that ECO-GNN algorithms are more effective in power optimization.
(3) Unlike commercial tools or previous works (see Section 2) that require multiple iterations to assign appropriate V_{th} types, our framework ECO-GNN only needs one-pass to determine the final V_{th} type for every design instance.
(4) Rather than treating our learning-driven framework as a blackbox, we implement a GNN-based explanation method [24] to quantitatively interpret the V_{th}-assignment predictions made by our framework. Given a target node, the method identifies the influential local sub-graph that has high contribution to its V_{th}-assignment.
(5) To the best of our knowledge, this is the first work that formulates signoff power optimization problem into a graph learning problem, and validates the proposed framework using an industrial-leading commercial tool under an advanced technology node.

2 RELATED WORKS AND MOTIVATIONS

The research in V_{th}-assignment for power optimization has been conducted extensively throughout the years. Previous works can mainly be categorized into the following streams:

- Non-Analytical Methods: This category includes methods that employ greedy-related [6, 10, 15], simulated annealing [5, 16], or dynamic programming [8, 9] algorithms to find feasible solutions. However, these approaches are highly sensitive to heuristics and are often design-specific. Therefore, they cannot be generalized to unseen designs or different technology nodes.
- Analytical Methods: Algorithms in this category often formulate the discrete sizing problem into the Convex optimization [17, 20] or Lagrangian optimization problem [8, 13, 18, 19] whose objective is to minimize the power under certain timing constraints. These methods are considered to yield better and more reliable optimization results than the non-analytical methods. However, solving an optimization problem using numerical approaches is extremely time-consuming, which makes these studies impractical for real-world usage.
- Machine Learning: Recently, machine learning emerges as a promising approach to solve the V_{th}-assignment problem with huge benefits in the optimization quality and runtime. The authors of [2] leverage linear regression to find feasible solutions based on path slack estimation. Another work [14] utilizes support vector machine (SVM) with lazy timing analysis to further enhance the optimization quality. However, these studies neglect that the final gate-type of each design instance highly depends on the characteristics of its neighbors. Therefore, the machine learning methods leveraged in these studies are not sufficient to perform the power optimization accurately without spending significant amount of time in feature engineering.

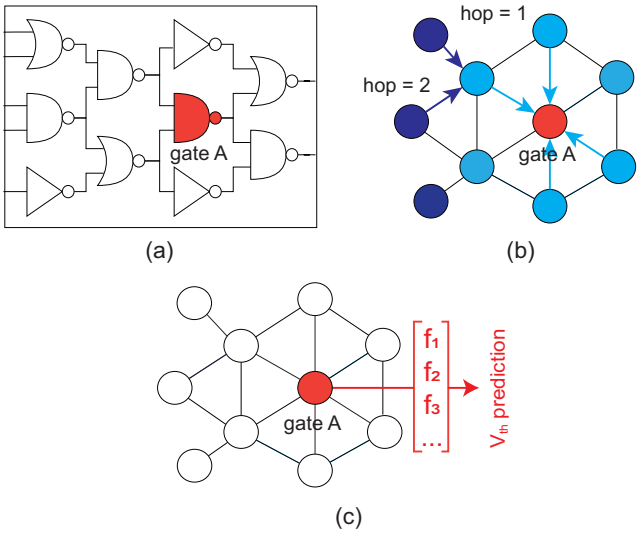

Figure 1: High-level view of our ECO-GNN framework, (a) input netlist, (b) graph representation learning, (c) V_{th} prediction. Note that (b) and (c) visualize the original netlist in a clique-based representation.

Apart from the specific drawbacks raised in each category aforementioned, there are several common drawbacks in all of the previous works. First, the timing models or constraints that they utilize are oversimplified and lack formal validations from commercial tools. Therefore, their methods cannot be extended for advanced technologies or industrial-scale designs. Second, the original ISPD-2012 benchmarks [12] that most of them leverage for evaluations are problematic. We analyze the benchmarks using *Synopsys PrimeTime*, and discover that the original worst negative slack values across all the designs range from $-1ns$ to $-8ns$, where all the target frequencies are less than $1GHz$. This simple fact makes previous works unrealistic, because the power optimization is meaningful only if the optimized designs are in signoff quality. Finally, none of the previous works interpret the optimization results achieved by their methods, where they all consider their optimization engines/models as blackboxes.

In this paper, we solve all the limitations and drawbacks raised above. We propose a transferable graph-learning-based signoff

power optimization framework with high fidelity as the commercial tool *Synopsy PrimeTime*. Furthermore, we leverage a GNN-based explanation method [24] to interpret the V_{th}-assignments made by our framework to ensure that our framework is reliable.

3 OVERVIEW OF ECO-GNN FRAMEWORK

Recently, GNNs have revolutionized many research areas, spanning from biology, social science, chemistry, and many others [4]. They perform effective graph representation learning, where the goal is to construct meaningful node embeddings that accurately characterize the nodes in the graph. In general, GNNs follow a message passing scheme, where a feature vector of a node can be considered as a message being iteratively transformed and passed to its neighboring nodes. At the end of the graph learning process, the initial node features are transformed into better representations that can be utilized in downstream tasks such as link prediction, node classification, and clustering [23].

Figure 1 presents a high-level view of our framework ECO-GNN. Since VLSI netlists can be naturally represented as hypergraphs, in this paper, we leverage a specific variant of GNNs named GraphSAGE [4] to conduct graph representation learning directly on the netlist graphs. After getting the learned representations, we utilize a softmax-based classification model to predict the V_{th}-assignments that optimize the signoff power. Note that the entire learning is an end-to-end process. The classification loss that represents the cross-entropy between our predictions and the ground-truths from Synopsys PrimeTime is utilized to update the parameters inside GNN and the classification model through gradient descent.

The detailed learning process shown in Figure 1 works as follows. Given an input netlist as shown in Figure 1(a), to determine the V_{th}-assignment of the target cell (red-colored), we first leverage a GNN to sample and aggregate the features from its neighboring cells as shown in Figure 1(b). Then, we predict its V_{th}-assignment based on the aggregated representation vector as shown in Figure 1(c).

The goal of this work is to construct a "general framework" that achieves *commercial-quality* signoff power optimization results *at inference time* (the testing time of the model). Note that our framework does not assume any pre-defined netlist structure, so it is generalizable to every design. After learning on a few designs, it has the facility to determine the V_{th}-assignments on the unseen ones that optimize the signoff power. The detailed algorithms of our framework are described in Section 5.

4 DESIGN OF EXPERIMENTS

In this work, we follow the experimental setting of the ISPD-2012 power optimization contest as many previous works [6, 10, 14–16, 18, 19], where all the cells in a given design are initially in the lowest V_{th} type (tightest timing constraint). As mentioned in Section 2, we re-synthesize the ISPD benchmarks using *TSMC 28nm* technology node to ensure all the designs are in signoff performance before conducting the power optimization through V_{th}-assignments.

4.1 Problem Formulation

Given a netlist $G = (V, E)$, where V denotes the instances in the design, and E represents the logical connections. Assume that for each instance $v \in V$, there are n V_{th}-assignments available from the standard cell libraries. Let $x_v^j = 1$ if instance v is realized with j-th V_{th} choice in the libraries and $x_v^j = 0$ otherwise. We formally define the signoff power optimization problem as follow:

$$\text{minimize} \sum_{i=1}^{|V|} \sum_{j=1}^{n} P(v_i^j) x_{v_i}^j, \tag{1}$$

where $P(v_i^j)$ represents the signoff power of instance v_i when j-th choice of V_{th}-assignment is realized such that the worst negative slack (WNS) along with the total negative slack (TNS) do not degrade after the assignments, and no new DRVs are added.

Table 1: 20 initial node features used in our GNN. We obtain them using an initial PPA analysis.

type	# dim.	description
max output slew	1	max transition of output pin
max input slew	1	max transition of input pin(s)
wst output slack	1	worst slack of output pin
wst input slack	1	worst slack of input pin(s)
output cap limit	4	max driving cap of output pin per V_{th}
max leakage	4	max leakage per V_{th}
tot input cap	1	sum of input pin cap
tot fanout cap	1	output net cap + input pin cap of fan-outs
tot fanout slack	1	sum of worst slack of fan-outs
wst fanout slack	1	worst. slack of fan-outs
avg fanin cap	1	average cap of fan-ins
wst fanin slack	1	worst slack of fan-ins
tot sibling cap	1	sum of input pin cap of siblings
tot sibling slack	1	sum of worst slack of siblings

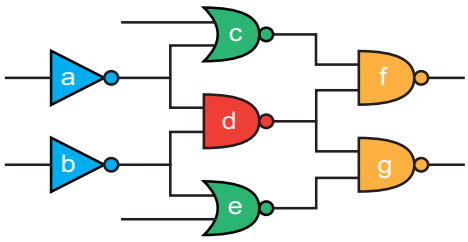

Figure 2: Initial feature construction of cell d, where its fan-ins $\{a, b\}$, siblings $\{c, e\}$, and fan-outs $\{f, g\}$ are taken into consideration.

4.2 Initial Node Features

Before leveraging GNN to conduct graph learning, we define an initial feature vector for each design instance as shown in Table 1. The term "initial" indicates that during the graph learning process, these original features are transformed to other representations that are more beneficial for the classification model to determine the appropriate V_{th}-assignments that optimzie signoff power.

Features in Table 1 are extracted from technology files, SPEF files, and timing reports. These 20 features are chosen based on domain knowledge and parameter sweeping experiments. Most of them are related to timing, because during the signoff power optimization, an instance's V_{th}-assignment changes only if the *WNS* and *TNS* do not degrade, and no DRV is introduced. Figure 2 further illustrates

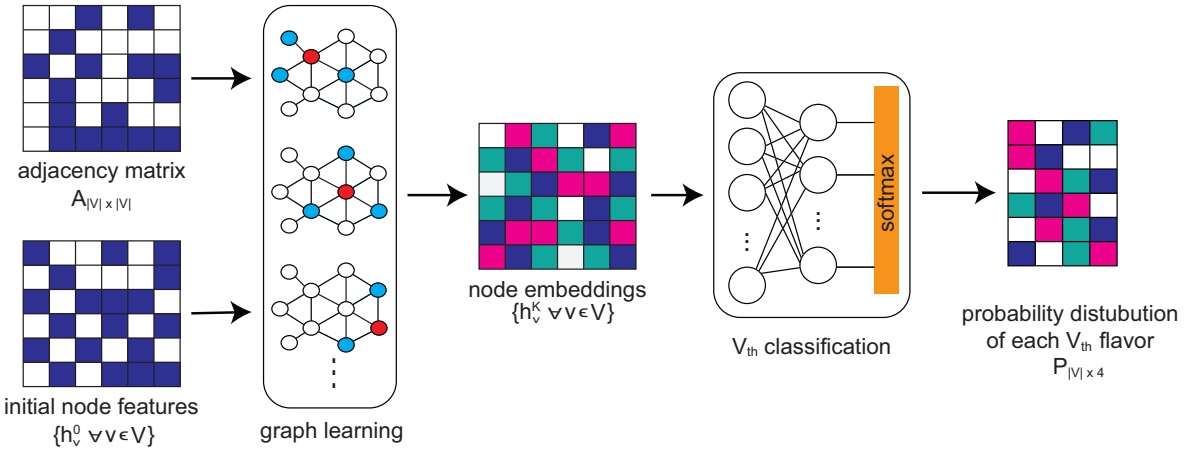

Figure 3: Illustration of our ECO-GNN learning process. The inputs include a netlist graph represented in an adjacency matrix A and its initial features h_v^0 defined in Table 1. First, we perform graph learning to generate the node embeddings that represent the netlist better than the initial features. Figure 4 provides details of the GNN structure used. Next, with the learned node embeddings, we conduct softmax-based classification to determine the final V_{th}-assignment that optimizes the signoff power.

Table 2: Dimension of matrices used in our work (see Figure 3). v denotes the number of gates in the circuit.

matrix	meaning	dimension
A	adjacency matrix of the netlist graph	$v \times v$
h_v^0	initial node features from PPA analysis	$v \times 20$
h_v^k	node embedding extracted by GNN	$v \times 128$
P	V_{th}-assignments from softmax function	$v \times 4$

the feature construction process. To determine the initial features of a target instance d, we take the information of its fanins (instances $\{a, b\}$), siblings (instances $\{c, e\}$), and fanouts (instances $\{f, g\}$) into account. However, these manually engineered features are not sufficient to predict the V_{th}-assignments that optimize the design signoff power. To get better node representations, we leverage GNNs to perform the graph representation learning.

5 ECO-GNN ALGORITHM

5.1 Overview of the Algorithm

Figure 3 shows a detailed illustration of the learning process in ECO-GNN framework. Given a netlist graph $G = (V, E)$, our framework first takes the initial node features defined in Table 1 as inputs. Then, it leverages GraphSAGE [4], a variant of GNNs to perform graph learning. The goal of graph learning is to obtain the node representations that better capture the underlying characteristics of the given netlist than the intial features. After graph learning, the learned representation vector of each node $v \in V$ is projected to a logit vector P_v through a softmax-based classification model, which is a neural network. The vector P_v represents the probability distribution of node v belonging to different V_{th} flavors that are available in the standard cell libraries.

Table 2 shows the size of matrices used in our framework. The adjacency matrix A represents the logical connections in the netlist, and the initial node features $\{h_v^0 \ \forall v \in V\}$ are the cell attributes

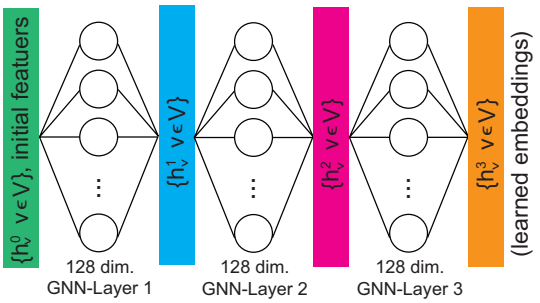

Figure 4: Our GNN architecture that maps the initial node features (20) into learned embedding features (128).

shown in Table 1. Note that the whole learning process, from graph learning to V_{th} classification, is end-to-end differentiable. Therefore, the parameters in the GNN and classification modules can be updated simultaneously using gradient descent.

5.2 GNN: Feature Aggregator

The goal of graph learning is to construct accurate node embeddings through effective feature aggregation. GNN functions as a feature aggregator that transforms the initial features h_v^0 for each node $v \in V$ into better representations h_v^K by *sampling and aggregating* the features within v's K-hop neighborhood. This aggregation process is performed iteratively, where for each hop $k \in \{1, ..., K\}$, a dedicated neural network (NN) $\mathbf{W}_k$ is developed to perform the transformation. These K dedicated NNs together form the GNN module in our framework as shown in Figure 4. Since the number of neighbors of a node scales exponentially as the hop-count increases, we fix the sampling size s_k at each hop k to improve the computational efficiency and to prevent overfitting.

Following the graph learning approach presented in [4], in this work, for each node $v \in V$, we obtain its representation vector h_v^k

at level[1] k by aggregating its representation h_v^{k-1} at the previous level with the features of its neighbors $N_k(v)$ sampled at k-hop as

$$h_{N_k(v)}^{k-1} = \text{maxpool}\left(\{\mathbf{W}_k^{agg} h_u^{k-1}, \forall u \in N_k(v)\}\right),$$
$$h_v^k = sigmoid\left(\mathbf{W}_k^{proj} \cdot \text{concat}[h_v^{k-1}, h_{N_v(v)}^{k-1}]\right), \quad (2)$$

where W_k^{agg} and W_k^{proj} denote the aggregation and projection matrices respectively, which together form the weights of the NN dedicated in *sampling and aggregating* features at the k-hop neighborhood. In the implementation, we set $k \in \{1, 2, 3\}$, and each NN (W_1, W_2, W_3) in the GNN module has an output dimension of 128. Note that the numbers 128 and 3 are chosen empirically based on parameter sweeping experiments.[2]

In summary, the initial feature vector h_v^0 for each node $v \in V$ is transformed to $h_v^{K=3}$ in R^{128}. The GNN model utilized in our framework can be considered as a "node filter", because it iterates through every design instance to find better node representations that can be utilized in the latter classification task of determining the V_{th}-assignments that optimize the design signoff power.

5.3 Loss Function

After leveraging GNN to perform graph representation learning, we take the learned node embeddings $\{h_v^K \in R^{128}, \forall v \in V\}$ as the inputs of our softmax-based classification model, which is a neural network, in order to determine the appropriate V_{th}-assignment for each design instance. As shown in Figure 3, the end of the classification model connects to a softmax function that outputs P, which is a $|V| \times n$ matrix denoting the probability of each node v belonging to n different V_{th} flavors, where $\forall v \in V$, $\sum_{c=1}^{n} P_{vc} = 1$. Note that n is limited to the discrete V_{th} values specified by the technology. The technology we utilize in this work is *TSMC 28nm* which has $n = 4$. A novelty of this work is that we map the discrete V_{th}-sizing problem into a multi-class classification problem, where the classification loss function is defined as:

$$\mathcal{L} = -\sum_{i=1}^{|V|}\sum_{c=1}^{n} Y_{ic} log(P_{ic}), \quad (3)$$

where $Y \in R^{|V| \times n}$ denotes the V_{th}-assignments made by the *Synopsys PrimeTime* ECO engine, which are taken as ground-truths. Essentially, our loss function (Equation 3) represents the cross-entropy between Y and P distributions. By minimizing Equation 3, we can update the parameters in the entire ECO-GNN framework.

5.4 Training Methodology

Algorithm 1 summarizes the training process. Lines 3–10 illustrate the *sampling and aggregating* process in graph learning, where for each node $v \in V$, we aggregate its neighboring features at each hop $k \in K$ through Equation 2. Note that before performing each aggregation, we normalize the node representations at previous level as shown in Line 2 and Line 9. This normalization accelerates the overall training process by reducing the oscillation of gradient descent. Based on the learned representation vectors, in Lines 11–15 we calculate the cross-entropy loss (Equation 3) from the softmax-based classification model, and leverage a gradient descent optimizer named *Adam* [7] to update the parameters in the framework by minimizing the loss function. The overall training process takes about 12 hours on the 9 training designs shown in Table 3 with a machine that has a 2.40 GHz CPU and a NVIDIA RTX 2070 graphic cards with 16 GB memory.

Algorithm 1 ECO-GNN training methodology.
We use default values of $K = 3, \alpha = 0.001, s_1 = 25, s_2 = 20, s_3 = 15, \beta_1 = 0.9, \beta_2 = 0.999$.

Input: (1) $G(V, E)$: netlist graph, (2) $A_{|V|\times|V|}$: adjacency matrix, (3) Y: tool optimization results, (4) n: number of available V_{th} flavors, (5) $\{h_v^0, \forall v \in V\}$: initial features. (6) K: depth of aggregation level, (7) $\{s_k, \forall k \in \{1, ..., K\}\}$: sampling size at k-hop neighborhood, (8) $\{\mathbf{W}_k, \forall k \in \{1, ..., K\}\}$: parameters of NN at hop k, (9) α: learning rate, (10) $\{\beta_1, \beta_2\}$: Adam parameters.
Output: $\{y\}$: learned node representations.

1: **while** $\{\mathbf{W}_k\}$ do not converge **do**
2: $h_v^0 \leftarrow \frac{h_v^0}{\|h_v^0\|_2}, \forall v \in V'$ ▷ initial features from Table 1
3: **for** $k \leftarrow 1$ to K **do**
4: **for** $v \in V'$ **do** ▷ sample and aggregate by Equation 2
5: $N_k(v) \leftarrow$ Sample s_k neighbors at k-hop
6: $h_{N_k(v)}^k = \text{maxpool}\left(\{\mathbf{W}_k^{agg} h_u^{k-1}, \forall u \in N_k(v)\}\right)$
7: $h_v^k = sigmoid\left(\mathbf{W}_k^{proj} \cdot \text{concat}[h_v^{k-1}, h_{N_v(v)}^k]\right)$
8: **end for**
9: $h_v^k \leftarrow \frac{h_v^k}{\|h_v^k\|_2}, \forall v \in V'$ ▷ reduce gradient oscillation
10: **end for**
11: **for** $v \in V'$ **do** ▷ minimize Equation 3
12: $p_v \leftarrow softmax(\mathbf{W}_k^{NN} \cdot f_v^K)$
13: $g_v \leftarrow \nabla_\theta[\sum_{c=1}^{n} Y_{ic} log(p_{vc})]$
14: $\{\mathbf{W}_k\} \leftarrow Adam(\alpha, \{\mathbf{W}_k\}, g_v, \beta_1, \beta_2)$
15: **end for**
16: **end while**

5.5 Complexity Analysis

The time complexity of ECO-GNN is linear with respect to the netlist size. Since the sampling size (s_k) at each aggregation level is constrained, GNN modules spend constant time in visiting every design instance and collecting features from its neighbors. Due to the large sparsity of the netlist adjacency matrix, we realize the adjacency matrix A shown in Table 2 in the compressed sparse row (CSR) format [22]. Therefore, the space complexity of ECO-GNN is pseudo-linear rather than quadratic with respect to the netlist size.

5.6 Handling Unseen Designs

A highlight of this work is that a trained ECO-GNN framework has the ability to perform commercial-quality signoff power optimization on *unseen* designs *at inference time*. This capability is independent of the netlist structure or the netlist size, because to determine the V_{th}-assignments that optimize the signoff power in

[1] Level is corresponding to the hop-count. When aggregating the features of a node at level k, the information within its k-hop neighborhood is considered.
[2] We varied them while monitoring the overall power saving vs. training time tradeoff. Due to the page limit, we omit the related experimental results. But, a general trend shows that the higher the values are, the more the power saving is at the cost of training time. But, the power saving saturates after some point.

an unseen design, we only need to take the initial features and the adjacency matrix as inputs, and ECO-GNN will determine the appropriate V_{th}-assignments through constant time inferencing. Unlike *PrimeTime* and previous works that require multiple iterations to determine the final V_{th}-assignments, our framework is a one-pass tool that generates tool-accurate results instantly.

5.7 Inner Workings of GNN Predictions

Unlike previous works who consider their optimization engines as blackboxes, in this paper, we implement a GNN-based explanation method [24] to interpret the V_{th}-assignment predictions made by our framework ECO-GNN. Given a set of target instances $\{v\} \in V$ in a netlist graph $G = (V, E)$, the goal is to find an influential sub-graph $G_S = (V_S, E_S)$ that has high contribution to the decision of $\{v\}$'s V_{th}-assignments. The objective of finding such sub-graph G_S can be quantitatively formulated as maximizing the mutual information (MI) between the original graph G and the sub-graph G_S as:

$$\max_{G_S} MI(G, G_S) = H(Y) - H(Y|G = G_S), \tag{4}$$

where $H(\cdot)$ denotes the entropy of the given distribution and Y represents the V_{th} prediction distribution of the target instances. Since $H(Y)$, the entropy of the prediction distribution based on the original graph, is a constant, maximizing Equation 4 is equivalent to minimizing the conditional entropy $H(Y|G = G_S)$ which can be formulated as:

$$H(Y|G = G_S) = -\mathbb{E}_{Y|G=G_S}\left[log\left(P_\theta(Y|G = G_S)\right)\right], \tag{5}$$

where θ denotes the parameters of the trained ECO-GNN framework. Note that due to the fact that the number of neighbors of the target nodes increases exponentially as the hop-count increases, in the implementation, we constrain G_S to search within the one-hop neighbors of the target instances $\{v\}$. In the context of the actual netlist, G_S represents the cells that are either the fanins, fanouts, or siblings of $\{v\}$ as well as the message passing flows (edge connectivities) that demonstrate how important features are aggregated. We believe this interpretability would give designers precious insights on what the framework has learned and whether the V_{th}-assignments are reliable or not.

6 EXPERIMENTAL RESULTS

In this section, we demonstrate the achievements of our ECO-GNN framework, which is implemented in *Python3* with *Tensorflow 1.0* library. We leverage 7 designs from the ISPD-2012 benchmark [12] and 7 other industrial designs to conduct the experiments. All 14 designs are synthesized under *TSMC 28nm* technology node by *Synopsys Design Compiler 2015*, and placed and routed using *Cadence Innovus v18.1*. To validate the signoff power optimization results of ECO-GNN, we use *Synopsys PrimeTime 2018* to perform timing and power analysis, and consider the *PrimeTime* ECO engine as the baseline across all experiments.

6.1 Benchmarks Details and Timing Corners

As mentioned in Section 2, due to the unrealistic nature of the ISPD-2012 benchmark that the worst negative slacks in the original designs range from $-1ns$ to $-8ns$, we re-implement all seven ISPD designs using *TSMC 28nm* technology node and commercial PD tools. Aside from the ISPD benchmarks, we introduce 7 other renowned industrial designs, including JPEG, TATE, LDPC, AES-128, NOVA, ECG from *OpenCores.org*, and RocketCore [1] which is a RISC-V-based multi-core system. To substantiate the generality of our framework, we utilize 9 designs in the training process, and perform the validations on the 5 *unseen* ones. The characteristics of these 14 designs are shown in Table 3.

Table 3: Our benchmarks and their attributes in *TSMC 28nm*.

Design Name	# Nets	# FFs	# Cells	Usage
RocketCore	93,812	16,784	90,859	
AES-128	90,905	10,688	113,168	
NOVA	138,171	29,122	136,537	
ECG	85,058	14,018	84,127	
LDPC	42,018	2,048	39,377	training
DMA	10,898	2,062	10,215	
PCI_BRIDGE	1,381	310	1,221	
DES_PERF	48,523	8,802	48,289	
B19	34,399	3,420	33,784	
TATE	185,379	31,409	184,601	
JPEG	231,934	37,642	219,064	
VGA_LCD	56,279	17,054	56,194	testing
LEON3MP	341,263	108,724	341,000	
NETCARD	317,974	87,317	316,137	

Following the experimental settings of the ISPD-2012 contest where all the designs are synthesized with one timing corner and one V_{th} flavor which has the tightest timing constraint, in this work, we synthesize all the designs using typical corner and ultra-low V_{th} flavor (tightest timing constraint) in *TSMC 28nm* for fair comparisons. In the *PrimeTime* ECO for signoff power optimization, each design instance is enabled to be swapped into one of the three other V_{th} flavors, which are low, high, and ultra-high types, or remain as the ultra-low type (4 choices in total). Therefore, the solution space of our V_{th}-assignment problem is $4^{|V|}$, which is almost impossible for designers to perform design exploration in an exhaustive manner.

6.2 Optimization Results on Unseen Designs

In this experiment, we compare the signoff power optimization results achieved by our framework ECO-GNN with the commercial tool *Synopsys PrimeTime*. To substantiate the generality of ECO-GNN, we only use 9 designs for training, and perform validations on the 5 unseen ones as shown in Table 3. Note that to perform meaningful and reasonable signoff power optimization, each design is originally implemented in signoff frequency, where the WNS is close to 0. The optimization constraints are that the WNS and TNS do not degrade and no violation is introduced after the optimization.

Table 4 demonstrates the optimization results. Compared with *PrimeTime*, ECO-GNN achieves up to 14X runtime improvement with similar optimization quality. Unlike previous works that do not utilize commercial signoff tools for validations, we demonstrate that our framework performs tool-accurate signoff power optimization without degrading the original signoff performance of each unseen design. Note that each design in Table 4 has different target frequencies, which proves that the optimization achieved is not confined by design characteristics. The inference time of ECO-GNN

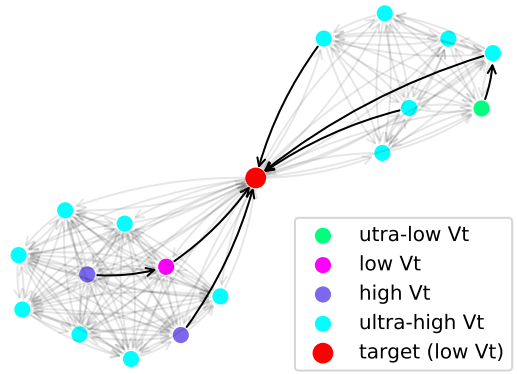

Figure 5: Graph learning explanation on b19 benchmark. The majority of the neighbors are ultra-high V_{th}, but cells with lower V_{th} types have higher importance to the target node. As a result, low V_{th} is assigned to the target node.

is measured on a machine with 2.40 GHz CPU and a NVIDIA RTX 2070 graphics card with 16GB memory, where *Synopsys PrimeTime* is ran on a machine with 2.50 GHz CPU and 8 cores enabled.

Table 4 also reports the micro F1-score as the evaluation metric of the classification task, owing to the fact that our framework ECO-GNN is performing supervised learning that we take the V_{th}-assignments from *Synopsys PrimeTime* as ground-truths in the training process. Note that micro F1-score represents the accuracy of multi-class classification. In the table, we observe that ECO-GNN performs the the V_{th}-assignments in high fidelity as *PrimeTime*.

Finally, due to the fact that V_{th}-assignments directly optimize the design leakage power, Figure 6 further shows the instance-based leakage power consumption maps of the unseen designs, which are corresponding to the optimization results presented in Table 4. In the figure, we compare the leakage power consumption of each instance in the original designs with the ones after using ECO-GNN to perform signoff power optimization. Across all designs, we observe that ECO-GNN effectively reduces the overall leakage power without introducing extra hotspots.

6.3 Discussion of Optimization Results

Power Perspective. As shown in Table 4, the optimizations through V_{th}-assignments achieved by our framework and *PrimeTime* improve both leakage power and total signoff power. This is because we follow the experimental settings from the ISPD-2012 contest [12] as many previous works [6, 10, 14–16, 18, 19]. The setting suggests all the cells to be in ultra-low V_{th} (tightest timing constraint) before the optimization. Therefore, for a design instance, a swap from ultra-low V_{th} to other V_{th} types in *TSMC 28nm* not only improves its static power (leakage) but also the dynamic power as the capacitance load is reduced.

Timing Perspective. As shown in the table, we observe that the WNS and TNS get improved as well. This comes from the fact that although *PrimeTime* will not upsize the V_{th} type of the cells that are on critical (negative slack) paths, the driving load of such cells may still be reduced if some of its fanout cells that are not on critical paths are swapped to higher V_{th} types, which in the end improves the overall timing as a by-product.

6.4 GNN Explanation

Instead of viewing our framework ECO-GNN as a blackbox, we validate our optimization results by explaining the V_{th}-assignments made by our framework. Figure 5 demonstrates the explanation results on the b19 design, where we plot the graph learning computational graph centered on the target node colored in red along with its neighbors using force-directed placement drawing [3]. Note that although we present single-instance explanation in this experiment for clarity, the proposed explanation method can be leveraged to perform the explanation of multi-instance as well.

To explain the V_{th}-assignment on the target node (red), we identify the important message passing flows within the local sub-graph. As mentioned in Section 5.7, we constrain the explanation method to search within the one-hop neighborhood. Therefore, every neighboring node in Figure 5 is either the fanin, fanout, or sibling of the target node. However, as shown in the figure, the influential features may not be passed directly from the neighbors to the target even though they are one-hop neighbors. This is because the message passing scheme in graph learning is bi-directional. For better illustration, in Figure 5, we plot two directed edges for each bi-directional edge in the graph learning computational graph to show how the influential features are being passed.

Figure 5 shows that the V_{th}-assignment made by ECO-GNN on the target node is reliable, because we observe that the final V_{th} type of the target node is more influenced by its minority neighbors who are in lower V_{th} types rather than the majority neighbors that are in the ultra-high V_{th} type. This aligns well with common design knowledge. Since cells in lower V_{th} types have larger capacitance, tighter constraints will be imposed on their drivers compared with cells in high-level V_{th} types. Therefore, we conclude that the V_{th}-assignment made by ECO-GNN on the target cell is reliable.

6.5 Why Does ECO-GNN Work?

In the experiments, we demonstrate that ECO-GNN achieves commercial quality signoff power optimization results with negligible runtime compared with *Synopsys PrimeTime*. The achievements of our framework can be accounted by two reasons. First, the initial modeling features (Table 1) accurately capture the underlying characteristics of each design instance that are related to the signoff power optimization. Specifically, the timing related features provide solid information for our framework to select appropriate V_{th}-assignments that optimize signoff power with the consideration of timing budget. Second, GNNs are highly powerful for solving the optimization problems on graphs. The final V_{th}-assignment of an instance highly depends on the information of its neighborhood structure. Therefore, unlike previous works [2, 14] who use traditional machine learning techniques to predict the V_{th}-assignment of an instance solely based on its handcrafted features, our framework acts as a graph filter that aggregates an instance's neighboring information to more accurately determine its final V_{th}-assignment through the classification model. Finally, with the validations from

Table 4: V_{th} re-assignment impact on power, timing, and runtime between ECO-GNN and *Synopsys PrimeTime*. Selected designs are *unseen* during training. Note that both leakage and total power reduce from V_{th} re-assignment, because our initial designs before ECO optimization are using ultra-low V_{th} only as suggested in [12]. Timing also improves because of the gate capacitance reduction from higher V_{th}.

Design	Target Frequency	Optimization Engine	Leakage Power (mW)	Total Power (mW)	WNS (ps)	TNS (ps)	Runtime (sec)	F1-Score ($micro$)
TATE	$1.2GHz$	Before Opt.	38.3	345.0	-2.4	-3.2	-	0.90
		PrimeTime	1.84	282.7	-0.5	-0.6	141	
		ECO-GNN	1.72	280.6	-0.9	-1.8	16 (9X)	
JPEG	$1.1GHz$	Before Opt.	57.5	376.8	-13.1	-228.7	-	0.85
		PrimeTime	3.4	294.6	-5.7	-69.6	120	
		ECO-GNN	3.8	296.9	-11.4	-182.4	15 (8X)	
VGA_LCD	$1.8GHz$	Before Opt.	18.0	212.7	-3.4	-14.1	-	0.89
		PrimeTime	3.7	184.6	-2.6	-4.4	69	
		ECO-GNN	3.5	183.3	-3.2	-11.8	5 (14X)	
LEON3MP	$700MHz$	Before Opt.	101.4	576.6	-16.3	-246.0	-	0.88
		PrimeTime	15.1	459.8	-8.4	-76.7	341	
		ECO-GNN	12.2	454.9	-12.8	-209.3	28 (12X)	
NETCARD	$1GHz$	Before Opt.	78.1	651.5	-2.4	-4.2	-	0.86
		PrimeTime	9.9	544.3	-0.8	-1.1	302	
		ECO-GNN	6.9	537.6	-1.2	-2.7	26 (12X)	

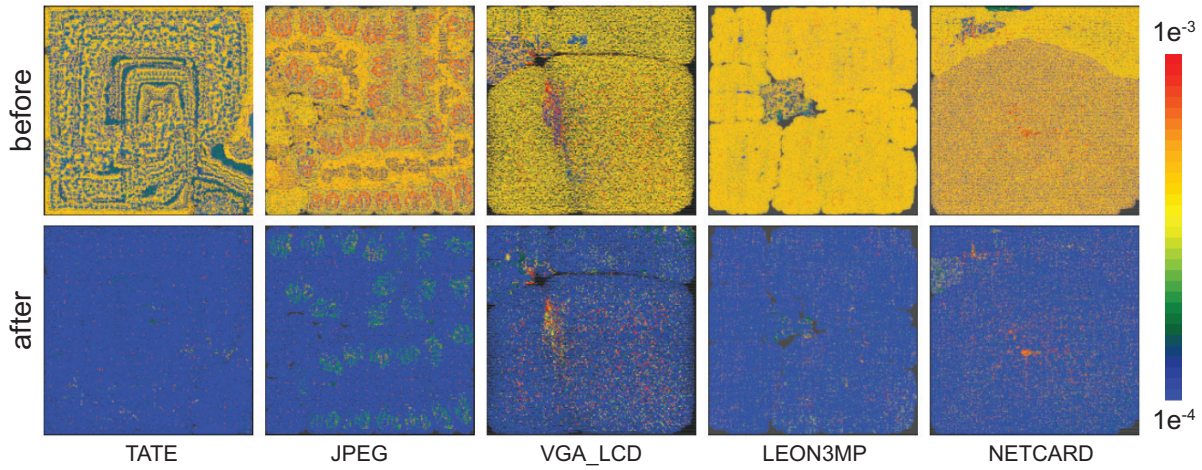

Figure 6: Leakage power consumption of each design instance before and after using ECO-GNN to perform optimizations. The designs are *unseen* during training, and the unit is mW.

the explanation method, we conclude that this work successfully presents a solution to the long lasting V_{th}-assignment problem.

In spite of the superior performance achieved, we still see some limitations of the proposed framework. We observe that *Synopsys PrimeTime* consistently delivers better timing results, and ECO-GNN does not consistently improve the signoff power from the commercial tool. In fact, this is resulted from the modeling errors occurred in the learning process. Although we take the V_{th}-assignments from *Synopsys PrimeTime* as the ground-truths, there always exists a gap between the predictions of our framework and the actual assignments made by the tool. Nonetheless, the goal of this work is *not to replace* commercial signoff tools, but to provide PD engineers a fast, accurate, and reliable estimation of the amount of power recovery to expect from the signoff tools.

7 CONCLUSION AND FUTURE WORK

We have proposed ECO-GNN, a transferable signoff power optimization framework that provides commercial-quality signoff power optimization results *instantly* on unseen designs. Furthermore, we have presented a GNN-based explanation method to demonstrate the reliability of our framework. In the future, we plan to validate ECO-GNN on other advanced technologies and to extend it to deal with more PD stages so that fast GNN-based optimization can be performed throughout the entire PD flow.

8 ACKNOWLEDGMENT

This work is supported by the National Science Foundation under Grant No. CNS 16-24731 and the industry members of the Center for Advanced Electronics in Machine Learning.

REFERENCES

[1] K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, et al. The rocket chip generator. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016.

[2] S. Bao. Optimizing leakage power using machine learning. CS Department, Stanford University, 2010.

[3] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. Software: Practice and experience, 21(11):1129–1164, 1991.

[4] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In Advances in neural information processing systems, pages 1024–1034, 2017.

[5] M. Hashimoto, H. Onodera, and K. Tamaru. A power optimization method considering glitch reduction by gate sizing. In Proceedings of the 1998 international symposium on Low power electronics and design, pages 221–226, 1998.

[6] J. Hu, A. B. Kahng, S. Kang, M.-C. Kim, and I. L. Markov. Sensitivity-guided metaheuristics for accurate discrete gate sizing. In Proceedings of the International Conference on Computer-Aided Design, pages 233–239, 2012.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[8] L. Li, P. Kang, Y. Lu, and H. 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. ACM, 2012.

[9] Y. Liu and J. 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.

[10] S. Mok, J. Lee, and P. Gupta. Discrete sizing for leakage power optimization in physical design: A comparative study. ACM Transactions on Design Automation of Electronic Systems (TODAES), 18(1):15, 2013.

[11] W. Ning. Strongly np-hard discrete gate-sizing problems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 13(8):1045–1051, 1994.

[12] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo. The ispd-2012 discrete cell sizing contest and benchmark suite. In Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design, pages 161–164. ACM, 2012.

[13] M. M. Ozdal, S. Burns, and J. Hu. Gate sizing and device technology selection algorithms for high-performance industrial designs. In Proceedings of the International Conference on Computer-Aided Design, pages 724–731. IEEE Press, 2011.

[14] S. Patanjali, M. Patnaik, S. 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.

[15] M. Rahman and C. Sechen. Post-synthesis leakage power minimization. In 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), pages 99–104. IEEE, 2012.

[16] T. Reimann, G. Posser, G. Flach, M. Johann, and R. Reis. Simultaneous gate sizing and vt assignment using fanin/fanout ratio and simulated annealing. In 2013 IEEE International Symposium on Circuits and Systems (ISCAS2013), pages 2549–2552. IEEE, 2013.

[17] S. Roy, W. Chen, C. C.-P. Chen, and Y. H. Hu. Numerically convex forms and their application in gate sizing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 26(9):1637–1647, 2007.

[18] S. Roy, D. Liu, J. Um, and D. Z. Pan. Osfa: A new paradigm of gate-sizing for power/performance optimizations under multiple operating conditions. In Design Automation Conference, page 129. ACM, 2015.

[19] A. Sharma, D. Chinnery, S. Bhardwaj, and C. 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.

[20] J. Singh, V. Nookala, Z.-Q. Luo, and S. Sapatnekar. Robust gate sizing by geometric programming. In Proceedings. 42nd Design Automation Conference, 2005., pages 315–320. IEEE, 2005.

[21] S. Sirichotiyakul, T. Edwards, C. Oh, and J. Zuo. Primetime user guide: Fundamentals, 2005.

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

[23] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2020.

[24] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnn explainer: A tool for post-hoc explanation of graph neural networks. arXiv preprint arXiv:1903.03894, 2019.