PowPrediCT: Cross-Stage Power Prediction with Circuit-Transformation-Aware Learning

Yufan Du[1,2*], Zizheng Guo[2,3*], Xun Jiang[2], Zhuomin Chai[2], Yuxiang Zhao[2], Yibo Lin[2,3,4†], Runsheng Wang[2,3,4], Ru Huang[2,3,4]
[1]School of EECS, Peking University [2]School of Integrated Circuits, Peking University
[3]Institute of EDA, Peking University [4]Beijing Advanced Innovation Center for Integrated Circuits
{nbsdyf,xunjiang,yuxiangzhao}@stu.pku.edu.cn,{gzz,yibolin,r.wang,ruhuang}@pku.edu.cn,zhuominchai@whu.edu.cn

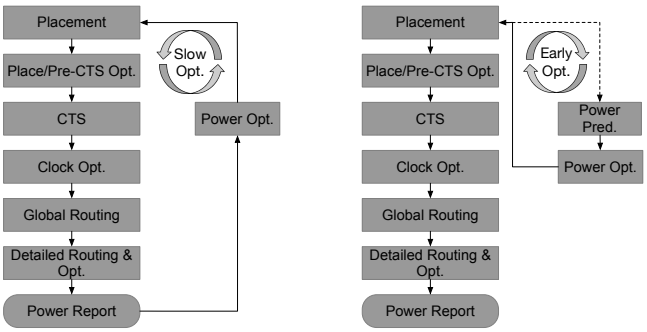
ABSTRACT

Accurate and efficient power analysis at early VLSI design stages is critical for effective power optimization. It is a promising yet challenging task to model the circuit power at early design stages, especially during placement with the clock tree and final signal routing unavailable. Additionally, optimization-induced circuit transformations like circuit restructuring and gate sizing can invalidate fine-grained power supervision. Addressing these difficulties, we introduce the first circuit-transformation-aware power prediction model at placement stage with robust generalization capabilities. Our technology includes a dedicated clock tree model and an innovative train-and-calibrate scheme that effectively integrates topological and layout features. Compared to the cutting-edge commercial IC engine Innovus, we have significantly reduced the cross-stage power analysis error between placement and detailed routing.

1 INTRODUCTION

Power is one of the most fundamental objectives in VLSI design optimization. Compared with performance and area costs, power footprint determines design quality in more intricate aspects, including energy efficiency, thermal performance, and voltage drop reliability. As a result, circuit power optimization, especially at early design stages, becomes essential as technology advancements continue to push physical and architectural limits.

Power optimization at early design stages such as placement can utilize a much larger design search space, which provides better trade-offs with other metrics such as timing. Early-stage power optimization requires fast early-stage power feedback to reduce costly design iterations (Figure 1). The accuracy of such feedback is directly tied to successful optimization. However, due to missing design information such as clock tree and signal routing, cutting-edge commercial tools [1] show up to 23% discrepancy in their early-stage power feedbacks, leading to mistargeted optimization efforts and eventual inferior power quality.



(a) Traditional flow (b) Proposed flow

Figure 1: (a) Obtaining accurate power analysis feedback is time-consuming due to the long and compute-intensive design stages. (b) Early-stage power prediction enables fast feedback and proactive power optimization.

Data-driven modeling such as machine learning (ML) is a promising methodology to circumvent missing design information across stages. ML-based early-stage analysis models have thus been applied to timing [2–4], voltage drop [5–7], routing congestion [8–10], etc. However, existing ML-based power prediction models [11–14] mainly focus on analysis speed-ups for some specific design stages with no cross-stage prediction available. Some of their model components also target architecture- or design-specific features, limiting their generalization to unseen designs.

On the other hand, building an accurate and generalizable power model that can provide cross-stage power feedback is quite challenging for the following reasons. (1) A remarkable portion (20–40%) of signoff power is consumed by the clock network, which is yet unavailable before clock tree synthesis (CTS) is completed. Modeling the CTS stage is known to be difficult due to the complexity of buffer insertions and skew management algorithms. (2) Modern IC tools conduct heavy netlist restructuring across stages to optimize design quality, which leads to mismatch (up to 21% [3]) of pins and nets between netlists from different stages. This makes it nontrivial to apply reasonable supervision on fine-grained power data. (3) Optimization iterations between stages, e.g., moving cells around, also introduce distribution shifts that degrade model performance.

This paper proposes the first cross-stage circuit-transformation-aware power prediction model with superior accuracy. Our model works at placement stage and generates accurate post-detailed-routing power prediction for all cells and nets. We propose various techniques to overcome the cross-stage challenges, including a dedicated graph neural network (GNN) model for clock tree modeling, a

* Equal contribution, † Corresponding author.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3657349

Table 1: Comparison of various power prediction methods.

Model	Main Analysis Scenario	Method	Cross-stage Aware[1]	Generality to Unseen Circuits[2]	Main Objective
McPAT [14]	RTL design	Static modeling	No	Yes	Pre-synthesis exploration
Aladdin [15]	RTL design	Execution-driven	Yes	Yes	Pre-synthesis exploration
MasterRTL [16]	RTL design	ML-based	Yes	Yes	Pre-synthesis exploration
SimplePower [17]	RTL design	LUT and Cluster	No	Yes	Analysis speed-up
PRIMAL [18]	RTL design	CNN	No	Yes	Analysis speed-up
GRANNITE [13]	Physical design	GNN	No	Yes	Analysis speed-up
ECO-GNN [19]	Signoff	GNN	No	Yes	Analysis speed-up
PowerTap [20], APOLLO [11]	Runtime	Proxy-based	No	No	Post-manufacture monitor
Innovus at PL[3]	Physical design	RC extraction and LUT	Yes	Yes	Backend cross-stage
PowPrediCT	Physical design	GNN and CNN	Yes	Yes	Backend cross-stage

[1] If power analysis considers potential transformation information in subsequent stages, it is cross-stage aware. [2] Based on the comparison presented in [21].
[3] Innovus performs power analysis at multiple stages, but our focus is on its placement stage results.

train-and-calibrate supervision scheme to combat netlist mismatch, and a layout-aware convolutional neural network (CNN) model for spatial information modeling and module-wise calibration. We summarize our key contributions as follows.

- To the best of our knowledge, this is the first ML-based cross-stage circuit-transformation-aware power prediction model with superior accuracy and generalization capability. Compared with the cutting-edge commercial tool Innovus, our method significantly reduces the switching power error between placement and post-routing stages from 23.4% to only **4.6%**, and the total power error from 9.6% to **2.0%** on unseen designs.
- Our model achieves remarkable runtime speed-up, 600× faster than the full flow and comparable with the original Innovus early power prediction.
- We propose a new train-and-calibrate scheme for training cross-stage analysis models. This scheme is *transformation-aware*, i.e., compatible with netlist restructuring and offers fine-grained supervision at cell and net levels.
- We propose a dedicated GNN model for clock tree power modeling before the actual CTS stage, unlocking the practicality of cross-stage power prediction.
- We introduce a CNN model to calibrate our GNN power model predictions. The CNN model offers multi-modality not only on routing layout features but also on coarse-grained module topology features. This enriches the model view and provides even further accuracy improvements.

Our general methodology is effective for early-stage power feedback and extendable to other cross-stage prediction tasks in a broad perspective. We open-source our model in the ML for EDA community[1]. The rest of this paper is organized as follows. Section 2 gives the preliminaries and problem formulation; Section 3 explains the detailed framework; Section 4 demonstrates the results; Section 5 concludes the paper.

2 PRELIMINARIES

2.1 Power Analysis

In VLSI design flow, power consumption is split into three components [22]: switching, internal, and leakage power.

(1) **Switching Power** is the power consumed during charging and discharging of interconnect capacitance for each net.

$$P_{\text{switch}} = 0.5 \times C_L \times V^2 \times F \times A, \tag{1}$$

where C_L denotes the net capacitive load, V is the voltage, F is the frequency, and A is the net switching activity. The switching activity A can be obtained either through vector-based or vectorless toggle rate analysis. The capacitive load C_L, on the other hand, is related to signal routing that is unavailable at early stages.

(2) **Internal Power** is the power consumed by a cell during signal transitions due to internal capacitances and short circuit current. It is determined by specific cell models, often in lookup tables (LUTs) indexed by input signal transition and output load.

(3) **Leakage Power** is the basic static power consumed by cells. It can be obtained from the library using cell input states.

2.2 Power Modeling Methods

Power modeling is an active research field and previous solutions usually fall into 4 categories by their orthogonal objectives and problem formulations, as we listed in Table 1. This work focuses on *backend cross-stage* power modeling. To enable early-stage power optimization, commercial tools like Innovus analyze early-stage power using classic methods with human knowledge. These methods utilize preliminary design and early global routing data but cannot often anticipate the impact of future circuit transformations and clock tree construction, leading to significant cross-stage power estimation discrepancies.

In addition to backend cross-stage modeling, some early-stage power prediction works such as McPAT [14] and Aladdin [15] operate at *pre-synthesis* stages. With quite limited design information available, their primary focus is to explore the architectural design space with limited power prediction accuracy.

Due to the long runtime of rigorous power analysis and circuit simulation, a lot of works have been proposed in recent years that focus on *analysis speed-up*. For example, works like ECO-GNN [19] have shown promising performance boosts in power calculation. PRIMAL [18] and GRANNITE [13] only provide toggle rate prediction and do not handle power model. Regarding analysis, they usually focus on a fixed design stage where the circuit graph has been mostly determined. As a result, they also don't need to consider cross-stage circuit transformations that might compromise accuracy with limited early-stage design information.

After completing circuit design and manufacturing, it is also necessary to dynamically monitor the power consumption *at runtime*. To this end, methods like APOLLO [11] and PowerTap [20] use a subset of signals as monitors to model the overall power consumption. This formulation does not need a generalizable model

[1] https://github.com/Yufan-Du/PowPrediCT/

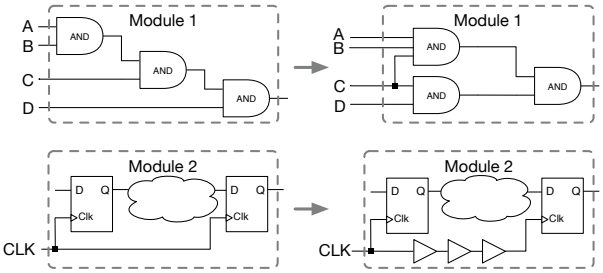


Figure 2: Examples of circuit transformation: logic restructuring and buffer insertion.

Table 2: Dataset statistics with remarkable cell transformations.

Design Name	#Pins	#Nets	#Cells	Cell Transformation (%)[1]
nvdla-small	1032047	285809	270072	16.00
Vortex-small	424710	124184	113961	18.87
Vortex-large	3753858	1105491	1018221	57.27
openc910-1	3065091	769380	754981	14.72
zero-riscy	138743	36233	35969	13.55
RISCY	180286	47241	46184	16.17
RISCY-FPU	252656	66911	65464	19.97

[1] The percentage of cells inserted, removed or altered in type from placement to post-routing stage.

for different circuit designs and requires training the model on a design-per-design basis.

2.3 Cross-Stage Circuit Transformation

In VLSI design flow, circuit transformations like netlist restructuring and gate sizing [22] are pivotal for design optimization, as shown in Figure 2. According to our observation, such transformations can result in up to 21% cell mismatches on our test circuits (Table 2) which is similar to results in [3]. Due to the transformations, a pin (resp. cell) at the early stage is not guaranteed to match a pin (resp. cell) in subsequent stages, as shown in Figure 2. This makes it difficult to present well-formed supervised learning problems. Although this can be mitigated by using endpoint features for timing prediction [3], such a technique cannot be applied to power analysis where the fine-grained power is concerned instead of only endpoint slacks.

2.4 Graph Neural Networks in Circuit Modeling

Graph neural networks (GNNs) have emerged as an innovative means of processing graph-structured data like circuit netlists [3, 13, 23]. They are designed to capture graphs' intricate relationships and structural patterns. As a simple illustration, a node's feature update in GNNs can be depicted as:

$$h_v^{(l+1)} = f\left(h_v^{(l)}, \text{AGGREGATE}\left(h_u^{(l)} : u \in \text{Neighbors}(v)\right)\right) \quad (2)$$

where $h_v^{(l+1)}$ represents the updated feature of node v, AGGREGATE is a function that combines the features of neighboring nodes, f combines information, $h_u^{(l)}$ is the feature of node u, and Neighbors(v) denotes the set of neighboring nodes of v.

While GNNs are inherently powerful, their basic form primarily captures information from immediate neighbors in each layer. Multiple GNN layers are required to embed features of nodes multiple hops away. This leads to a particularly challenging training process, often resulting in difficulty achieving convergence in large circuits with deep logic levels [24].

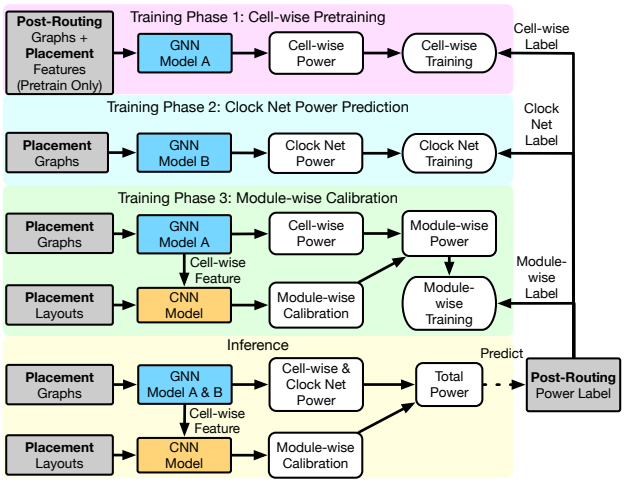


Figure 3: Framework of our approach.

2.5 Problem Formulation

Problem. Given a design at placement stage, our objective is to efficiently predict the post-routing total power consumption with high accuracy and generalization capability. This prediction should take into account circuit transformations, like clock tree construction, final signal routing patterns, and potential optimizations.

3 METHODOLOGY

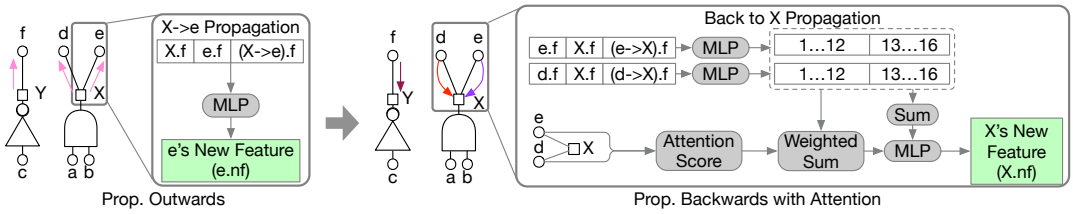
3.1 Flow Overview

Our power prediction framework is trained in 3 phases. We present an overview of the framework in Figure 3.

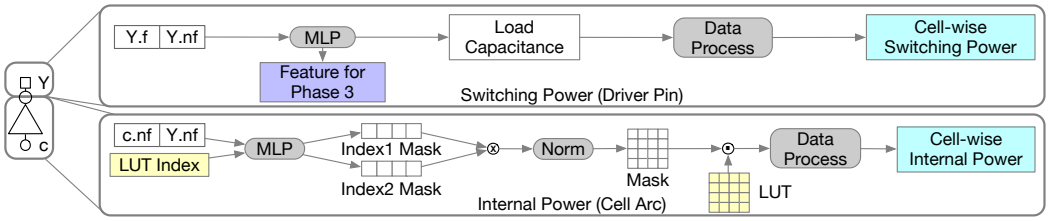
Phase 1: Cell-wise GNN Model Pretraining. We first pretrain a cell-wise power prediction GNN model (A in Figure 3) on post-routing circuit graphs with placement features using the training designs. Pretraining without circuit transformation helps the model learn the basics of power prediction without being affected by transformation glitches. The dependency on post-routing graphs will later be eliminated in Phase 3 where we calibrate the model to placement graphs.

Phase 2: Clock Net Power Prediction. The clock net requires dedicated handling due to its extremely high fanout (before CTS) and special routing patterns. We use a dedicated GNN model (B in Figure 3) to predict the power of the clock net, which we train using placement graphs (i.e., no CTS information used) as features and power labels after CTS and routing as targets.

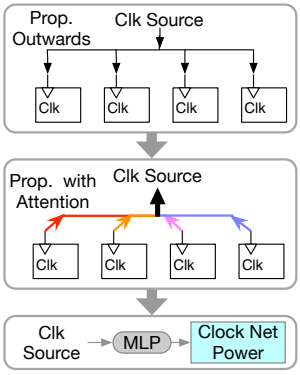
Phase 3: Module-wise Calibration. In this stage, we take the pretrained GNN model A and replace its input with placement graphs. Extra epochs update GNN model A to recognize circuit transformations between placement and routing while maintaining its power prediction capability. In addition, we use a CNN to model the layout features and utilize the natural grouping of cells in Verilog modules to calibrate the overall model performance further.



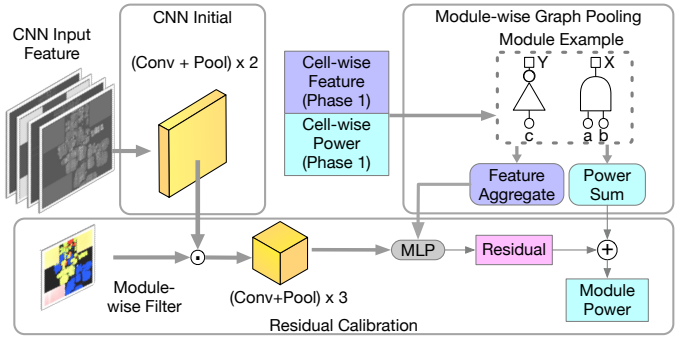
(a) Information propagation (Phase 1).



(b) Power analysis (Phase 1).



(c) Clock net power prediction (Phase 2).



(d) Module-wise calibration (Phase 3).

Figure 4: Overview of the algorithm phases.

3.2 Cell-wise GNN Model Pretraining

3.2.1 Circuit Graph Representation. In our graph-based model, nodes represent pins and edges represent cell arcs and driver-sink net arcs. The model inputs include:

List of node features: drive strength (1-hot), sequential element flag (marking sequential elements like flip-flops), fanout count, pin capacitance, rise/fall slews (obtained from early-stage timing analysis), and design flow-related parameters (design frequency and resource utilization).

List of net edge features: half perimeter wire length (HPWL) of the net, distance from source to this sink pin, and relative X and Y distance (ratios of Δx and Δy to total distance d, indicating edge orientation).

List of cell edge features: the non-linear power model (NLPM) look-up table values and indices.

3.2.2 Information Propagation. In our GNN model, the single information propagation operation is executed through a two-step process. Initially, during the edge message generation step, each edge creates a message based on its features and the nodes it connects. Following this, during the node message aggregation step, nodes accumulate and process these messages, integrating information received from all the edges that target them.

Our methodology actualizes this propagation twice through a bidirectional propagation approach, as shown in Figure 4(a):

Driver-to-Load Propagation: Each load node receives and directly adopts information from its corresponding edge message.

Load-to-Driver Propagation: Building upon the basic information propagation operation, our model introduces an enhanced attention mechanism for this step. It assigns variable weights to edges, enabling the driver pin node to selectively aggregate signal information from load pin nodes. This mechanism focuses on crucial load pins in large nets, enhancing the model's ability to learn about potential net routing scenarios.

3.2.3 Power Analysis. Following graph propagation, we utilize node and edge features for power analysis, as shown in Figure 4(b).

Internal Power Prediction: Leveraging the features after graph propagation, we use an ML-based approach to construct a mask for the internal power LUT, similar to [24]. The mask, when dot-multiplied with the LUT, facilitates an interpolation-like process. The resulting values are post-processed, including multiplication with the equivalent toggle rate for each cell arc input and k-factor [22].

The input pin toggling's impact on internal power follows a similar principle and is therefore not elaborated further.

Switching Power Prediction: For net power analysis, we utilize enriched driver pin features after the graph propagation. These features are processed and subsequently multiplied by the net toggle rate input. Additionally, each driver pin outputs a feature vector, which serves as an input for Phase 3 processing.

3.3 Clock Net Power Prediction

Our GNN-based clock power model, as shown in Figure 4(c), has a similar structure to the bidirectional GNN propagation model used in Phase 1. We predict the clock tree power by considering the interaction between the clock root and all register sinks. We incorporate the attention mechanism in the message aggregation procedure during clock modeling. This allows the model to recognize register clusters and different clock tree levels during feature propagation by assigning them with correlated attention vectors according to the locations.

3.4 CNN-aided Module-wise Calibration

Our GNN model, effective in various aspects, still has limitations that CNN integration aims to address in three key areas.

Enhanced Net Representation: The GNN model, while capturing basic spatial features, struggles with complex interactions like coupling capacitance and congestion effects. By integrating CNNs, we achieve a more comprehensive net representation, incorporating detailed spatial features such as cell density and congestion, crucial for accurately analyzing routing scenarios.

Circuit Transformation Prediction: The spatial view of the CNN model can give complementary hints to GNN on a general trend of spatial-induced circuit transformations, e.g., the ones introduced due to routing congestion, cell density balance, and macro interactions [3].

Reduced Cross-stage Variance: Although circuit transformations can cause cell mismatch, the Verilog module hierarchy is much more stable. As our CNN focuses on module-level features, a more robust supervision target can be achieved.

Therefore, we utilize a CNN model with inputs including cell density, early congestion, macro positioning, and RUDY (Rectangular Uniform wire DensitY). As depicted in Figure 4(d), the workflow first integrates convolution and pooling layers to process these layout features. It then focuses on cell regions within each module through module-specific masks. Subsequently, additional convolution, pooling, and MLP layers incorporating aggregated cell-wise features from Phase 1 are applied. This results in residuals used subsequently to calibrate results obtained from Phase 1.

3.5 Discussion on Early-Stage Data Integration

Cross-stage ML-based analysis models can choose to start their performance by integrating the outputs of commercial tool early-stage predictions as input features, as suggested in [25, 26] for timing and area analyses. In practice, we also found it helpful to integrate this feature for all types of power analysis. Leakage power benefits the most, potentially because it has more consistent patterns across stages. For internal power and switching power, we added the corresponding feature sets.

4 EXPERIMENTAL RESULTS

4.1 Setup

We implement our models using PyTorch and the DGL graph learning framework [27]. The experiments are conducted on a Linux machine with an NVIDIA A100 GPU (40GB), a 10-core Intel Xeon Processor (Skylake, IBRS), and 72GB memory.

Our dataset (Table 2) is derived from CircuitNet [28], consisting of 7 netlists implemented using a commercial 14nm FinFET technology. Each design is implemented at 2 operating frequencies (200MHz and 500MHz) and 4 layout utilizations, creating 8 samples per design. In our experimental setup, we perform cross-validation by training on 6 designs and testing on the remaining unseen one. The test result is the average of 8 samples from the unseen design. We train our model for 30 epochs in Phase 1 and an additional 2 epochs in both Phases 2 and 3. The overall training is completed within 1 hour. Our model targets generalization across different netlists. As a result, we do not account for training runtime in total runtime evaluation.

4.2 Power Prediction Accuracy

Table 3 lists a complete cross-validation accuracy comparison between Innovus and our PowPrediCT model at placement stage. We set the golden results as reported by Innovus after detailed routing. Early-stage Innovus analysis exhibits notable discrepancy with later stages, yielding an average 9.652% total power error and 23.375% switching power error. On the other hand, our model outperforms Innovus on all but the smallest designs, yielding an average 1.981% total power error and 4.559% switching power error.

To further demonstrate the effectiveness of the proposed PowPrediCT model, we compare its performance with a vanilla GNN model and a simplified PowPrediCT model without calibration and clock tree modeling (Phase 1 only). The vanilla GNN employs a semi-supervised learning approach that treats mismatched cells as unlabeled instances. It unsurprisingly yields poor accuracy on designs with extensive transformations. Meanwhile, the "Phase 1 Only" column in Table 3 shows notable degradations in accuracy compared to full PowPrediCT. These results demonstrate the effectiveness of our proposed model in cross-stage power prediction and the importance of modeling the clock tree and handling circuit transformations during the prediction.

4.3 Runtime Comparison

The runtime of power evaluation feedback is critical for a fast design closure because early stages involve multiple optimization iterations. A signoff accurate power feedback requires a full implementation process, including the time-consuming CTS and routing stages. This often takes several hours as shown in Table 4, which is unacceptable to use as early stage feedbacks. On the other hand, our model inference only takes 1.31 seconds on average. Even considering data preparation runtime, we achieve very small total power error with hundreds of time speed-up compared to running full-chip implementation. The Innovus built-in power prediction algorithm runs in 2 minutes on average. Integration of our flow can add up to 1 minute to that runtime, which mostly comes from the data preparation step. The slower data preparation is mainly due to the Tcl interface overhead, which can be greatly optimized given better data integration with the placement engine in future production.

Table 3: Relative error (%) comparison. The best results are highlighted in bold.

Design name	Internal Power		Switching Power		Leakage Power		Total Power			
	Innovus at PL	PowPrediCT	Innovus at PL	PowPrediCT	Innovus at PL	PowPrediCT	Vanilla GNN	Innovus at PL	Phase 1 Only	PowPrediCT
RISCY	**0.651**	0.808	24.999	**3.144**	4.533	**0.747**	8.322	9.202	2.961	**1.285**
RISCY-FPU	1.246	**0.910**	23.120	**2.495**	4.181	**0.820**	6.770	9.475	2.795	**1.467**
Vortex-large	1.567	**0.955**	9.626	**8.774**	9.972	**5.469**	29.892	5.100	9.540	**4.334**
Vortex-small	1.455	**1.169**	15.442	**1.287**	5.640	**4.410**	31.880	7.691	8.697	**0.991**
nvdla-small	2.139	**0.356**	33.709	**7.294**	7.315	**2.686**	9.370	10.892	3.457	**2.009**
openc910-1	4.014	**0.734**	29.568	**5.870**	4.026	**1.146**	5.123	15.122	6.692	**2.599**
zero-riscy	0.896	**0.365**	27.166	**3.049**	5.449	**1.288**	7.683	10.083	1.597	**1.179**
Average	1.710	**0.757**	23.376	**4.559**	5.874	**2.367**	14.149	9.652	5.106	**1.981**

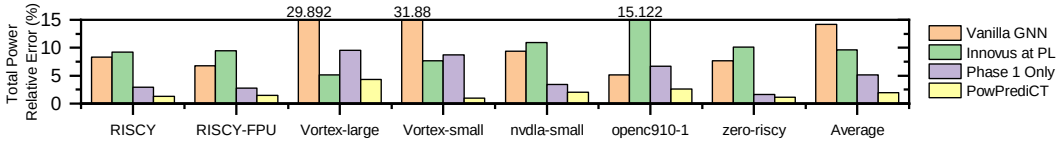


Figure 5: Total power relative error of different methods across different test sets.

Table 4: Runtime (s) comparison. Column names represent: PA (Power Analysis), Pre. (Data Preprocessing), Infer (Inference).

Design	Innovus at PL	Innovus Full Flow				PowPrediCT		
	PA at PL	CTS	Routing	PA	Total	Pre.	Infer	Total
RISCY	37	2034	7254	34	9322	17.61	0.72	**18.33**
RISCY-FPU	51	7949	9625	43	17617	21.23	0.65	**21.88**
Vortex-large	217	82328	19031	107	101466	156.94	2.68	**159.62**
Vortex-small	92	10092	12605	78	22775	36.56	1.55	**38.11**
nvdla-small	219	49669	22019	105	71793	110.01	0.95	**110.96**
openc910-1	242	13112	50569	145	63826	117.27	2.31	**119.58**
zero-riscy	26	1728	5523	27	7278	12.72	0.31	**13.03**
Average	126	23845	18089	77	42011	67.48	1.31	**68.79**

5 CONCLUSION

In this work, we present an accurate and efficient cross-stage power prediction model for power feedback at placement stage. We use a dedicated GNN model to predict the power of the clock tree before CTS. We present a transformation-aware training and calibration scheme compatible with netlist restructuring. We combine the capabilities of both topological features and layout features and both fine-grained and coarse-grained power annotations. The experimental results show that our generalizable PowPrediCT model drastically reduces cross-stage power analysis error compared to a commercial tool on unseen designs. Moreover, it achieves significant speed-up relative to the tool's full flow. Our future work includes integration into open-source placement engines. We believe our powerful power model can further empower early-stage power optimization.

ACKNOWLEDGE

This work was supported in part by the National Science and Technology Major Project (Grant No. 2021ZD0114702), the Natural Science Foundation of Beijing, China (Grant No. Z230002), and the 111 Project (B18001).

REFERENCES

[1] "Cadence Innovus Implementation System," http://www.cadence.com.
[2] X. He *et al.*, "Accurate timing prediction at placement stage with look-ahead RC network," in *Proc. DAC*. San Francisco California: ACM, Jul. 2022, pp. 1213–1218.
[3] Z. Wang *et al.*, "Restructure-tolerant timing prediction via multimodal fusion," in *Proc. DAC*, 2023, pp. 1–6.
[4] Y. Ye *et al.*, "Fast and Accurate Wire Timing Estimation Based on Graph Learning," in *Proc. DATE*. Antwerp, Belgium: IEEE, Apr. 2023, pp. 1–6.
[5] J.-X. Chen *et al.*, "Vector-based Dynamic IR-drop Prediction Using Machine Learning," in *Proc. ASPDAC*. Taipei, Taiwan: IEEE, Jan. 2022, pp. 202–207.
[6] Z. Xie *et al.*, "PowerNet: Transferable Dynamic IR Drop Estimation via Maximum Convolutional Neural Network," in *Proc. ASPDAC*. Beijing, China: IEEE, Jan. 2020, pp. 13–18.
[7] C.-T. Ho, A. B. Kahng, and E. Departments, "IncPIRD: Fast Learning-Based Prediction of Incremental IR Drop," in *Proc. ICCAD*. ACM, 2019.
[8] B. Wang *et al.*, "LHNN: lattice hypergraph neural network for VLSI congestion prediction," in *Proc. DAC*. San Francisco California: ACM, Jul. 2022, pp. 1297–1302.
[9] Z. Xie *et al.*, "RouteNet: routability prediction for mixed-size designs using convolutional neural network," in *Proc. ICCAD*. San Diego California: ACM, Nov. 2018, pp. 1–8.
[10] R. Kirby *et al.*, "CongestionNet: Routing Congestion Prediction Using Deep Graph Neural Networks," in *Proc. VLSI-SoC*. Cuzco, Peru: IEEE, Oct. 2019, pp. 217–222.
[11] Z. Xie *et al.*, "Apollo: An automated power modeling framework for runtime power introspection in high-volume commercial microprocessors," in *Proc. MICRO*, 2021, p. 1–14.
[12] D. Zoni, L. Cremona, and W. Fornaciari, "Powerprobe: Run-time power modeling through automatic rtl instrumentation," in *Proc. DATE*, 2018, pp. 743–748.
[13] Y. Zhang, H. Ren, and B. Khailany, "Grannite: Graph neural network inference for transferable power estimation," in *Proc. DAC*, 2020, pp. 1–6.
[14] S. Li *et al.*, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. MICRO*, 2009, pp. 469–480.
[15] Y. S. Shao *et al.*, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *Proc. ISCA*, 2014, pp. 97–108.
[16] W. Fang *et al.*, "Masterrtl: A pre-synthesis ppa estimation framework for any rtl design," in *Proc. ICCAD*, 2023.
[17] W. Ye *et al.*, "The design and use of simplepower: a cycle-accurate energy estimation tool," in *Proc. DAC*, 2000, pp. 340–345.
[18] Y. Zhou *et al.*, "Primal: Power inference using machine learning," in *Proc. DAC*, 2019, pp. 1–6.
[19] Y.-C. Lu *et al.*, "Eco-gnn: Signoff power prediction using graph neural networks with subgraph approximation," *ACM TODAES*, vol. 28, no. 4, may 2023.
[20] D. Zoni *et al.*, "Powertap: All-digital power meter modeling for run-time power monitoring," *Microprocess. Microsystems*, vol. 63, pp. 128–139, 2018.
[21] J. Zhai *et al.*, "Mcpat-calib: A risc-v boom microarchitecture power modeling framework," *IEEE TCAD*, vol. 42, no. 1, pp. 243–256, 2023.
[22] Cadence Design Systems, *Innovus User Guide*, Cadence Design Systems, Inc., 2022, version 21.13.
[23] K. Zhu *et al.*, "Exploring logic optimizations with reinforcement learning and graph convolutional network," in *Proc. DAC*, 2020, pp. 145–150.
[24] Z. Guo *et al.*, "A timing engine inspired graph neural network model for pre-routing slack prediction," in *Proc. DAC*. ACM, 2022.
[25] A. Agiza *et al.*, "Graphsym: Graph physical synthesis model," in *Proc. ICCAD*, 2023.
[26] K. Chang *et al.*, "DTOC: integrating Deep-learning driven Timing Optimization into the state-of-the-art Commercial EDA tool," in *Proc. DATE*. IEEE, 2023.
[27] M. Wang *et al.*, "Deep graph library: Towards efficient and scalable deep learning on graphs," *CoRR*, vol. abs/1909.01315, 2019.
[28] Z. Chai *et al.*, "CircuitNet: An open-source dataset for machine learning in vlsi cad applications with improved domain-specific evaluation metric and learning strategies," *IEEE TCAD*, 2023.