

Enabling Cross-Design Power Trace Prediction with GNNs for Gate-Level Netlists

Shih-Chun Lin¹, Yung-Chih Chen^{1,2}, and Bo-Hao Huang¹

¹Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

²Arculus System Co., Ltd., Hsinchu, Taiwan

{m11207442, ycchen.ee, m11307411}@mail.ntust.edu.tw

Abstract—Accurate cycle-by-cycle power estimation plays a critical role in the early stages of chip design, facilitating power, performance, and area (PPA) optimization. Recently, machine learning (ML)-based methods have emerged as faster alternatives to traditional electronic design automation (EDA) tools. However, they often require model retraining for each new design, which limits their general applicability and efficiency during early-stage design exploration. To address this, we propose a graph neural network (GNN)-based estimator for gate-level cycle-based power prediction, designed to achieve cross-design generalization. By exploiting the GNN’s ability to capture circuit structure and encoding standard cell types from the design library into node embeddings, our model effectively generalizes to unseen circuit designs without retraining. Experimental results demonstrate that our GNN-based estimator achieves over $29\times$ faster cycle-based power estimation than commercial EDA tools, with NRMSE below 3.37% and 5.19% for zero-delay and SDF-delay scenarios, respectively.

I. INTRODUCTION

As modern circuit designs continue to grow in complexity and scale, the associated specifications and constraints have become increasingly stringent. Consequently, efficient and accurate evaluation during the design exploration phase is essential. However, the expanding scale of contemporary designs makes circuit analysis increasingly computationally intensive and time-consuming. In particular, cycle-based power analysis, which evaluates power consumption at each clock cycle, is a critical step in early-stage chip design. It plays an important role in applications such as power-aware design optimization, pattern selection, and IR-drop and thermal analysis, all of which require fine-grained, time-resolved power information.

Commercial tools are typically employed to perform power analysis at the register-transfer level (RTL) or gate-level abstraction. Gate-level power analysis is more accurate than RTL analysis but comes at the cost of significantly higher runtime. Furthermore, gate-level power analysis can be categorized into cycle-based, which evaluates power on a per-cycle basis, and average, which computes power from toggle rates aggregated over the entire simulation period. Fig. 1 illustrates the typical workflow for gate-level power analysis using commercial tools, showing cycle-based analysis in Fig. 1(a) and average analysis in Fig. 1(b). The key difference lies in how switching activities are processed: cycle-based analysis extracts detailed per-cycle

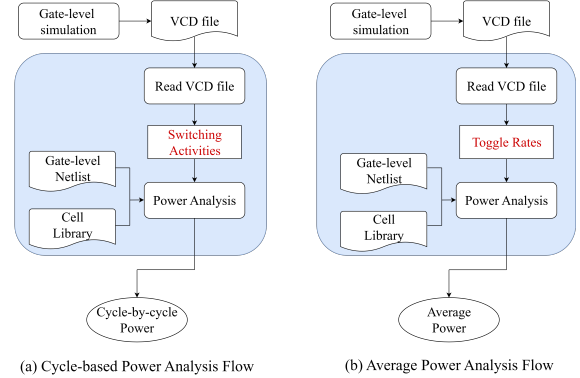


Fig. 1. Power analysis methods: (a) Cycle-based, using per-cycle switching activities; (b) Average, using aggregate toggle rates.

activity, whereas average analysis uses aggregate toggle rates. Consequently, cycle-based analysis can be hundreds to thousands of times slower than average power estimation.

In recent years, rapid advances in machine learning (ML) have encouraged the application of ML techniques to various problems in EDA [1]. While traditional EDA tools are optimized for accuracy, ML-based methods can offer substantial speedups in certain phases of the design flow, especially during early design exploration or iterative optimization.

However, despite these advances, existing ML-based models remain limited in their ability to generalize across designs for cycle-based power prediction [2]–[4]. The main limitation arises from existing methods that encode model input data based on the number of registers or cells, leading to varying input dimensions across different circuit designs. To overcome this limitation, we propose a GNN-based power estimator specifically designed to achieve cross-design cycle-based power prediction. Using the standard cell types directly extracted from the cell library, our method encodes circuit information in a way independent of design-specific dimensions, enabling our model to generalize effectively across different circuit designs. Additionally, by leveraging the GNN’s ability to capture detailed circuit structure and incorporating carefully defined key features related to power consumption, our model accurately predicts power for unseen designs.

The primary contributions of this paper are summarized as follows:

- We propose a GNN-based predictor for gate-level cycle-based power analysis with detailed and fine-grained esti-

This work was supported by the National Science and Technology Council, Taiwan, under grants NSTC 111-2221-E-011-137-MY3 and 113-2640-E-011-003.

mations.

- The model generalizes across designs, eliminating retraining needs and significantly saving computational resources and design iteration time.
- It supports both zero delay and SDF delay models, enabling accurate estimation under realistic timing.
- Our method significantly outperforms commercial EDA tools like PrimeTime PX [5], [6] in runtime.
- The model trained on small circuits maintains good accuracy when applied to larger, unseen designs.

The rest of this paper is organized as follows: Section II presents the preliminaries of this work. Section III reviews related works. Section IV presents our proposed method. Section V details our experimental setup and results, and Section VI concludes this work.

II. PRELIMINARIES

A. Components of Power Consumption

Power consumption consists of dynamic power and leakage power. The dynamic power is further composed of switching power and internal power.

1) *Leakage Power*: Leakage power is the static power consumed whether or not the circuit is switching states. It primarily arises from leakage currents flowing through transistors, even when they are turned off.

2) *Switching Power*: In cycle-based analysis, switching power arises when a cell output transitions within a cycle, typically from a low to a high logic level, thereby charging the associated load capacitance. It is given by the formula:

$$P_{switching} = \alpha \times C \times V^2 \times f \quad (1)$$

where C is the load capacitance, V is the supply voltage, f is the operating frequency, and α is the activity factor, representing the average number of output transitions per cycle.

Switching power is the dominant component of dynamic power consumption in digital circuits.

3) *Internal Power*: Internal power is the power consumed within a cell when any of its pins change state. This power is determined based on the internal switching activity of the gate, which depends on the states of pins other than the switching pin.

B. Delay Models in Power Analysis

Power analysis can be conducted under different timing assumptions, which significantly affect the accuracy and complexity of the analysis. They can be categorized into three major types:

1) *Zero Delay (ZD)*: ZD simulations ignore all delays, assuming an idealized circuit in which signals propagate instantaneously. The primary purpose of ZD analysis is to perform purely functional validation, without considering real-world timing effects.

2) *Unit Delay (UD)*: UD simulations assign an identical, uniform delay (typically one simulation time unit) to each logic gate or cell. This delay model captures basic timing relationships; however, it does not reflect accurate timing behavior in realistic scenarios.

TABLE I
COMPARISON OF RELATED WORKS AND OUR METHOD.

Work	Model	Inference Phase	Main Goal	Transfer.
PRIMAL [2]	MLP, CNN, XGBoost	RTL sim. (ZD)	GL Cycle-based power (ZD)	X
GRANNITE [10]	GNN	RTL sim. (ZD) + GL netlist	GL Average power (ZD)	V
PowPrediCT [11]	GNN	GL sim. (pre-route) + GL netlist	GL Average power (post-route)	V
AL-Power [3]	RNN-Encoder + DNN	RTL sim. (ZD) + GL sim. (SDF)	GL Cycle-based power (SDF)	X
CAPEDL [4]	Encoder + MLP	GL sim. (ZD)	GL Cycle-based power (ZD)	X
Ours	GNN	GL sim. (ZD or SDF) + GL netlist	GL Cycle-based power (ZD or SDF)	V

3) *Standard Delay Format (SDF)*: SDF simulations use detailed delay information that captures gate-level and interconnect delays on a per-net basis. Although highly accurate, SDF-based simulations require significantly longer computational time compared to ZD and UD models.

In this work, we focus on the gate-level netlist stage and implement our method using two commonly employed delay models: ZD and SDF, to capture both idealized and realistic timing conditions.

C. Overview of GNN

GNN are broadly categorized into spectral-based and spatial-based methods. Spectral-based approaches, such as GCNs [7], [8], rely on graph Laplacian eigendecomposition and operate in the spectral domain. They typically lack generalizability across varying graph topologies. In contrast, spatial-based methods like GraphSAGE [9] aggregate neighbor features directly in the node domain, supporting dynamic graphs and mini-batch training, making them more practical for large-scale applications. Due to their flexibility and scalability, spatial-based GNNs have become the dominant choice in recent research.

In this work, we adopt a spatial-based GNN to capture signal interactions through neighbor message passing, thereby improving efficiency and enhancing cross-design generalization. Moreover, since leakage and internal power are highly dependent on the logic states of a cell's side pins, the spatial-based GNN propagates information from fanin cells along directed edges, providing the per-pin state context required for accurate power estimation.

III. RELATED WORKS

ML methods have recently been leveraged to improve the efficiency and accuracy of power estimation in early design stages. For example, PRIMAL [2] introduced a comprehensive ML-based framework for cycle-based power estimation, employing multiple ML models including multilayer perceptrons (MLP), convolutional neural networks (CNN), and gradient boosting (XGBoost). PRIMAL utilizes switching activity traces extracted from the RTL simulation to predict gate-level (GL) power consumption at each cycle. Although PRIMAL improves efficiency over traditional methods, it cannot generalize across different designs and requires retraining for each new circuit.

GRANNITE [10] proposed a GNN-based method to achieve transferable average power estimation. It represents circuit

structures as graphs, captures node relationships and interactions, and uses GNN to propagate features effectively. It takes the state probabilities and toggle rates of primary inputs (PIs) as input and performs topological propagation to update node features, enabling it to predict the toggle rates of all cells in the circuit. A commercial tool is then used to calculate the average power based on the predicted toggle rates. Although GRANNITE achieves transferability, its application is limited to average power estimation without addressing the challenge of cycle-based power prediction.

PowPrediCT [11] is a cross-stage, circuit-transformation-aware power prediction framework that operates at the placement stage. It utilizes GNN and CNN to process placement-stage circuit data and predict the average power after detailed routing. However, PowPrediCT is limited to average power estimation and does not support cycle-based power prediction.

AL-Power [3] addressed the limitations of large training data requirements in ML-based methods by integrating active learning strategies and a recurrent neural network (RNN)-based autoencoder. This approach efficiently selects representative training samples, significantly reducing the amount of required labeled data while maintaining estimation accuracy. The method demonstrates notable performance improvements in both accuracy and computational efficiency. However, it fails to generalize to new circuits.

CAPEDL [4] introduced a cycle-based power estimation framework that employs an autoencoder for feature dimension reduction and an MLP for power prediction. It achieves low prediction error by capturing the nonlinear relation between gate-level switching activities and power consumption, but like prior works, CAPEDL fails to address the cross-design challenge. This is mainly because the number of cells varies across designs, leading to mismatched input feature dimensions.

While prior approaches have advanced power prediction, none provide a predictor that both generalizes to unseen designs and supports cycle-level accuracy. GRANNITE [10], for example, achieves cross-design transferability by leveraging topological propagation through the circuit, but its sequential node updates scale poorly with circuit size, and it predicts only average power. In contrast, our message-passing GNN updates all nodes in parallel, leverages GPU parallelism, and directly predicts cycle-level power, improving both efficiency and generalization.

Table I provides a summarized comparison of related works and our method. It highlights key aspects such as model architecture, inference conditions, primary estimation target, and whether the approach supports cross-design transferability.

IV. PROPOSED METHOD

A. Model Training and Testing

Fig. 2 illustrates the overall workflow, consisting of training and testing phases that share the same preprocessing but operate on different datasets with distinct purposes. In training, labeled data with accurate cycle-by-cycle power values are used to optimize model parameters through supervised learning. In testing, the trained model is evaluated on unseen circuits to

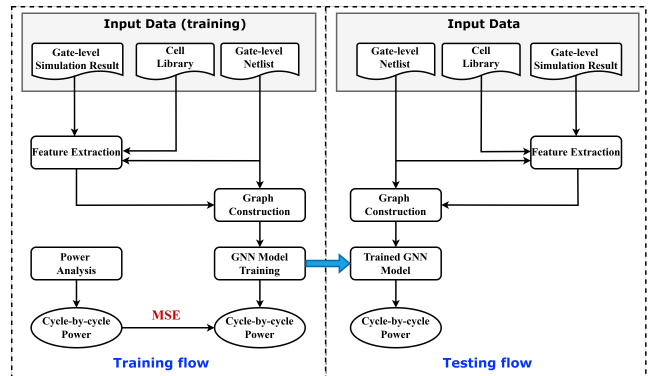


Fig. 2. Overall workflow.

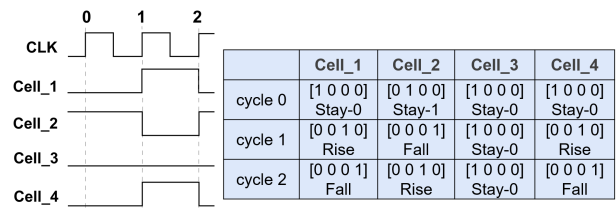


Fig. 3. Representation of switching activity using four-state one-hot encoding.

assess prediction accuracy and generalization. In both phases, gate-level simulation results, the standard cell library, and the gate-level netlist are used to extract features. A graph is then constructed from the netlist, with features embedded and passed through the model for training or inference.

B. Feature Extraction

The features utilized in our model can be categorized into four types: ① **Four-State Switching Activity**, ② **Glitch Count**, ③ **Probabilities of State 0/1**, and ④ **Total Load Capacitance of Rising Cells**. Among them, features ② and ③ are only used under the SDF scenario to characterize more realistic timing effects.

1) *Four-State Switching Activity*: Fig. 3 illustrates our method for encoding cell switching activity with a four-state one-hot representation. At each cycle, the activity is classified as stay-0, stay-1, rise, or fall, producing a 4-dimensional one-hot vector for each cell per cycle, as shown below:

- Stay-0: The cell remains in logic 0 \rightarrow [1 0 0 0]
- Stay-1: The cell remains in logic 1 \rightarrow [0 1 0 0]
- Rise: The cell switches from 0 to 1 \rightarrow [0 0 1 0]
- Fall: The cell switches from 1 to 0 \rightarrow [0 0 0 1]

We extract this feature of each cell from the gate-level simulation results.

2) *Glitch count*: Glitches are unintended transient signal transitions within a logic cell, typically caused by mismatched arrival times of input signals. Although short-lived, these glitches significantly impact the total dynamic power consumption. In our proposed model, the Glitch Count feature captures the number of glitches occurring within each cell during each simulation cycle. This feature is also derived directly from the gate-level simulation results.

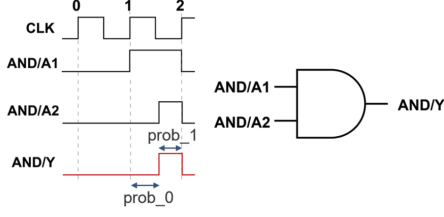


Fig. 4. Illustration of signal behaviors under SDF scenario.

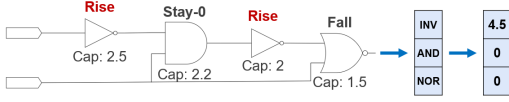


Fig. 5. Calculation of total load capacitance for cells with rising transitions, categorized by cell types.

3) *Probabilities of state 0/1*: Due to varying delays introduced in the SDF model, each cell may remain in different logic states (0 or 1) for varying durations within a single cycle. As illustrated in Fig. 4, signals exhibit distinct state-duration characteristics under the SDF model. Because the power consumption of a cell depends on its logic state, the probabilities of states 0 and 1 (prob_0 and prob_1) of each cell within each cycle are recorded as features. Incorporating these probabilities enables our model to capture the dynamic variations in power consumption caused by state-duration differences, thereby improving the fidelity of power estimation under realistic timing conditions. This feature is also extracted from the gate-level simulation results.

Note that the above-mentioned features of each cell are subsequently processed by a GNN layer, as detailed in a later sub-section.

4) *Total Load Capacitance of Rising Cells*: Switching power arises when a signal makes a transition from low to high, charging the load capacitance. Fig. 5 illustrates an example where two INV cells have rising transitions, with a total load capacitance of 4.5. For each cycle, we record all the cells with rising transitions, group them by standard cell type, and sum their load capacitances per type. Please note that cells with the same function but different sizes are treated as distinct types.

The load capacitance information is derived from the standard cell library and the gate-level netlist. Unlike the other three features, this one is combined with the GNN output embedding and fed into an MLP for further processing.

C. Model Structure

As illustrated in Fig. 6, the model first converts the gate-level netlist into a graph, where nodes represent cells and edges denote direct interconnections. A GNN layer processes node-level features, including Four-State Switching Activity, Glitch Count, prob_0 , and prob_1 . For each node, the GNN performs message passing by aggregating information from its fanin neighbors. The resulting node embeddings are then refined through a fully connected (FC) layer to enhance feature expressiveness. These refined representations are grouped by cell types to form an output embedding, which

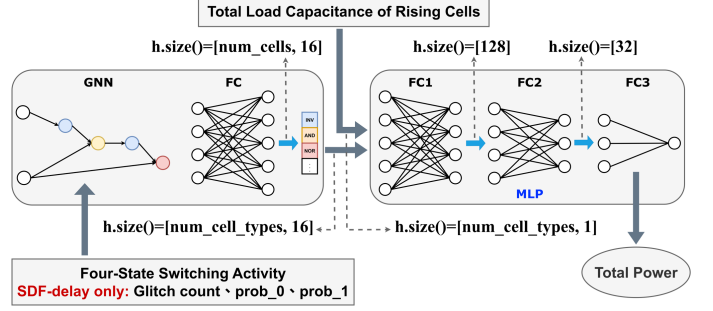


Fig. 6. Architecture of the proposed GNN-based power prediction model.

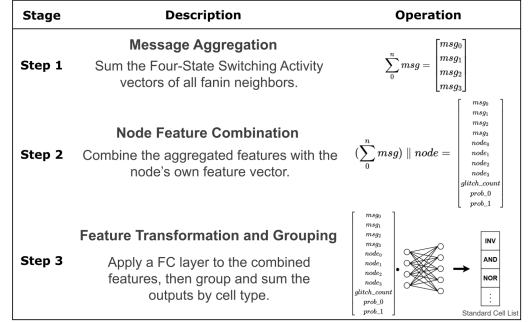


Fig. 7. Three-step GNN message passing mechanism.

is concatenated with the Total Load Capacitance of Rising Cells feature. The final feature vector is input to an MLP to predict power consumption.

Fig. 7 details the message passing process within the GNN layer. This process comprises three steps: (1) each node aggregates feature vectors from its fanin neighbors; (2) the aggregated features are combined with the node's own feature vector; and (3) the combined features are transformed through an FC layer. The FC layer is configured to output a 16-dimensional vector, producing a 16-dimensional representation for each cell. These outputs are then grouped and summed by cell types, yielding a fixed-size embedding of dimension $\text{num_cell_types} \times 16$.

Referring back to Fig. 6, this output embedding is then concatenated with the Total Load Capacitance of Rising Cells feature, which adds one additional scalar per cell type. Therefore, the final input to the MLP becomes a vector of dimension $\text{num_cell_types} \times (16 + 1) = \text{num_cell_types} \times 17$. The MLP consists of three FC layers with output dimensions of 128, 32, and 1, respectively. Leaky ReLU activation functions are applied after each layer except for the final output layer. The model finally produces a single scalar value representing the predicted power consumption for a cycle.

By keeping the dimensions of both the GNN output embedding and the Total Load Capacitance of Rising Cells feature fixed, determined solely by the number of cell types, this consistent structure enables robust cross-design generalization.

TABLE II
BENCHMARKS FOR TRAINING AND TESTING.

	Designs	Cell count range
Training set	c2670, c3540, c5315, c6288, c7552, adder, bar, max, sin, sqrt	295 ~ 11,519
Testing set	square, multiplier, log2, div, hyp	13,854 ~ 153,574

V. EXPERIMENTAL RESULTS

A. Experimental Setup

Our experiments were conducted on a workstation with a 3.9GHz 64-core Intel Xeon Gold 6526Y processor, an NVIDIA L40S GPU, and 512GB RAM. RTL designs were synthesized into gate-level netlists using Synopsys Design Compiler [12] with the SAED EDK 32nm standard cell library, which also generated corresponding SDF files for delay annotation. Each netlist was then simulated in Synopsys VCS [13] with 20,000 random input patterns to extract cycle-based switching activities, and power analysis was performed using PrimeTime PX [5], [6] to obtain accurate cycle-by-cycle power as ground truth. Both ZD and SDF models were evaluated.

The benchmark circuits were selected from the ISCAS'85 [14] and EPFL [15] suites, as summarized in Table II. In Experiments 1 and 2, we evaluated our method under ZD and SDF scenarios, respectively. Of the 15 circuits, 10 smaller ones were used for training and 5 larger ones for testing. Leveraging the model's cross-design generalizability, a single trained model was used to evaluate all test circuits. In Experiments 3 and 4, we re-implemented CAPEDL [4] and AL-Power [3], which are cycle-based power prediction models under ZD and SDF scenarios, respectively, for comparative evaluation.

Our model was implemented in PyTorch [16] and trained for 30 epochs using mean squared error (MSE) loss. We used the Adam optimizer with a learning rate of 5×10^{-5} and a weight decay of 0.01. Evaluation metrics include Normalized Root Mean Squared Error (NRMSE) and average power error, defined as follows:

$$NRMSE = \frac{1}{\bar{y}} \sqrt{\frac{\sum_i^n (y_i - \hat{y}_i)^2}{n}} \quad (2)$$

$$Average\ Power\ Error = \left| \frac{\bar{y} - \bar{y}_{pred}}{\bar{y}} \right| \quad (3)$$

where y_i and \hat{y}_i are the predicted and ground truth powers at cycle i ; n is the total number of simulation cycles; and \bar{y} and \bar{y}_{pred} are the average ground truth and predicted powers across all cycles, respectively.

Compared to average power error, NRMSE is more informative for the cycle-based power prediction task, as it captures the per-cycle deviation between the predicted and ground truth power values, reflecting both accuracy and stability over time.

B. Experiment 1: Cross-Design (ZD)

Experiment 1 evaluated the cross-design generalization capability of our model under the ZD scenario. As shown in Fig. 8, our model generalizes well to unseen and larger designs. The largest errors are observed for the div benchmark, with

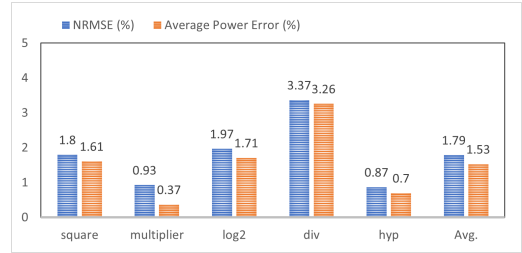


Fig. 8. Power prediction results under ZD scenario (Experiment 1).

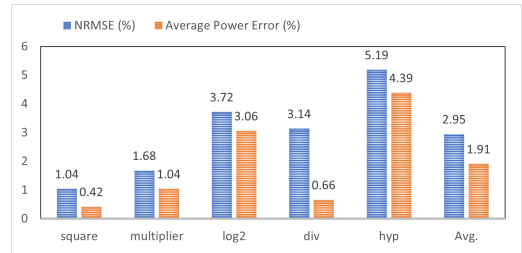


Fig. 9. Power prediction results under SDF scenario (Experiment 2).

an NRMSE of 3.37% and an average power error of 3.26%. Additionally, by leveraging GPU parallelism for GNN message passing, our method significantly outperforms PrimeTime PX [5], [6] in runtime efficiency. As presented in Table III, the speedup becomes more pronounced on larger circuits and reaches at least $29.77\times$, demonstrating the practicality of our approach for fast power analysis.

C. Experiment 2: Cross-Design (SDF)

Experiment 2 followed the same setup as Experiment 1, but incorporated realistic timing effects through SDF annotation. Fig. 9 demonstrates that our model remains effective when applied to larger and unseen designs under the SDF scenario, yielding prediction errors below 5.19% (NRMSE) and 4.39% (average power error). In terms of performance, the model exhibits notable runtime advantages over PrimeTime PX [5], [6], particularly as circuit complexity increases. As shown in Table III, it achieves a speedup of at least $67.30\times$ across all evaluated circuits, underscoring its suitability for rapid and scalable power analysis in practical design scenarios.

D. Experiment 3: vs. CAPEDL (ZD)

In Experiment 3, we compared our method against the re-implemented CAPEDL [4] under the ZD scenario. Since CAPEDL does not inherently support cross-design generalization, a separate model was trained for each of the five testing circuits using 16,000 random patterns and evaluated on 4,000 random patterns.

In contrast, our approach uses a single model trained on all 16,000 patterns from each circuit and evaluated individually on the 4,000 random patterns of each circuit. The results, shown in Table IV, indicate that our model outperforms CAPEDL in terms of NRMSE, achieving significantly lower errors across all designs, with values consistently below 1.28%. Although slightly worse in average power error, both models achieve errors below 1%, indicating strong predictive reliability.

TABLE III
SPEEDUP OVER PRIME TIME PX FOR 20,000 CYCLES UNDER ZD (EXP. 1)
AND SDF (EXP. 2) SCENARIOS.

Designs	Exp. 1			Exp. 2		
	Time (s)		Speedup (×)	Time (s)		Speedup (×)
	PTPX	Ours		PTPX	Ours	
square	322.41	10.83	29.77	834.47	12.40	67.30
multiplier	625.43	10.84	57.70	1598.12	11.78	135.66
log2	584.15	10.72	54.49	1576.47	13.17	119.70
div	939.53	10.97	85.65	1583.12	13.21	119.84
hyp	5985.84	30.50	196.26	12362.05	38.88	317.95
Avg.	1691.47	11.14	84.77	3590.85	17.89	152.09

TABLE IV
COMPARISON OF ACCURACY METRICS BETWEEN OUR MODEL AND
CAPEDL [4] (EXPERIMENT 3).

Design	NRMSE (%)		Avg. Power Error (%)	
	Ours	CAPEDL	Ours	CAPEDL
square	0.79	3.72	0.45	0.26
multiplier	1.05	2.86	0.41	0.23
log2	1.28	3.07	0.24	0.18
div	1.11	3.02	0.38	0.27
hyp	0.66	2.04	0.27	0.14
Avg.	0.98	2.94	0.35	0.22

E. Experiment 4: vs. AL-Power (SDF)

In Experiment 4, we compared our method against the re-implemented AL-Power [3] under the SDF scenario. The experimental setup follows that of Experiment 3. Note that although AL-Power uses active learning to reduce training data, we ensured a fair comparison by training both models on the same dataset. Table V presents the experimental results, showing that both models performed well, but our model achieved superior results in both NRMSE and average power error metrics.

In summary, the results of Experiments 3 and 4 highlight the effectiveness and practical advantage of our method over existing ML-based power estimation methods.

F. Ablation study

We evaluate feature effectiveness under the SDF scenario, using the same training/testing splits as Experiment 2. To isolate feature contributions, we vary the feature set while aggregating all five testing circuits into a single evaluation. We adopt Mean Absolute Percentage Error (MAPE) and the coefficient of determination (R^2) as metrics:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{\hat{y}_i} \right| \quad (4)$$

$$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2} \quad (5)$$

where \bar{y} is the average ground truth power across all cycles; y_i and \hat{y}_i are the predicted and ground truth powers at cycle i ; and n is the total number of simulation cycles.

The results, summarized in Table VI, show a clear progression: using only four-state switching activity yields poor accuracy (MAPE = 35.78%, $R^2 = 0.67$); adding glitch count markedly improves performance (MAPE = 10.87%,

TABLE V
COMPARISON OF ACCURACY METRICS BETWEEN OUR MODEL AND
AL-POWER MODEL [3] (EXPERIMENT 4).

Design	NRMSE (%)		Avg. Power Error (%)	
	Ours	AL-Power Model	Ours	AL-Power Model
square	1.70	2.58	0.15	0.16
multiplier	1.74	2.11	0.17	0.16
log2	2.12	2.18	0.44	0.75
div	1.93	1.87	0.11	0.37
hyp	0.93	2.35	0.08	0.53
Avg.	1.68	2.22	0.19	0.39

TABLE VI
COMPARISON OF FEATURE SETS IN TERMS OF MAPE AND R^2 .

Feature Set	MAPE (%)	R^2
Only 4-states	35.78	0.67
4-states + glitch count	10.87	0.96
4-states + prob_0/prob_1	25.65	0.85
Complete features	2.49	0.99

$R^2 = 0.96$); adding prob_0/prob_1 also helps (MAPE = 25.65%, $R^2 = 0.85$); combining all features achieves the best outcome (MAPE = 2.49%, $R^2 = 0.99$). For intuition, Fig. 10 visualizes R^2 across configurations, highlighting the incremental gains from each feature.

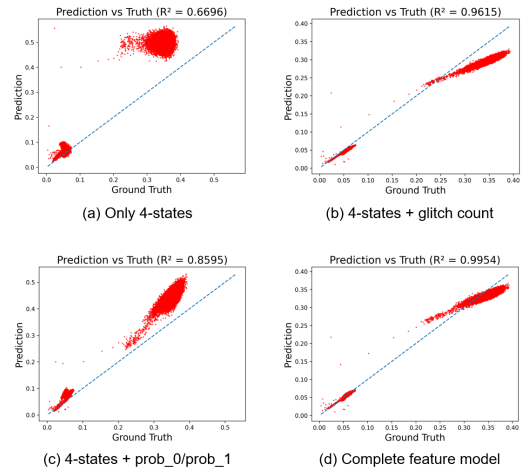


Fig. 10. Visualization of prediction accuracy for different feature sets in terms of R^2 under the SDF scenario.

VI. CONCLUSION

This paper proposes a GNN-based method for efficient and accurate cycle-by-cycle gate-level power estimation, with a focus on cross-design generalizability. By embedding standard cell types into node representations, the model maintains consistent input dimensionality across designs. Experimental results demonstrate high accuracy under both ZD and SDF scenarios, significant runtime advantages over PrimeTime PX, and improved NRMSE compared to state-of-the-art ML approaches such as CAPEDL and AL-Power, while maintaining competitive average power error. Training on smaller designs and evaluating on larger unseen circuits highlights the model's scalability and practical applicability.

REFERENCES

- [1] L. Chen, Y. Chen, Z. Chu, W. Fang, T. Ho, Y. Huang, S. Khan, M. Li, X. Li, Y. Liang, Y. Lin, J. Liu, Y. Liu, G. Luo, Z. Shi, G. Sun, D. Tsaras, R. Wang, Z. Wang, X. Wei, Z. Xie, Q. Xu, C. Xue, E. F. Y. Young, B. Yu, M. Yuan, H. Zhang, Z. Zhang, Y. Zhao, H. Zhen, Z. Zheng, B. Zhu, K. Zhu, and S. Zou, "The dawn of ai-native EDA: promises and challenges of large circuit models," *CoRR*, vol. abs/2403.07257, 2024.
- [2] Y. Zhou, H. Ren, Y. Zhang, B. Keller, B. Khailany, and Z. Zhang, "Primal: Power inference using machine learning," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2019.
- [3] S.-M. Liu, S.-Y. Fang, H.-W. Chang, M.-C. Lee, and P. Wei, "Active learning-based practical power estimation considering multi-cycle paths," in *2024 25th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, 2024.
- [4] T. Liu, H. Zhao, and Y. Lyu, "Capedl: Cycle-accurate power estimation with deep learning," in *2024 2nd International Symposium of Electronics Design Automation (ISED)*, pp. 642–647, 2024.
- [5] Synopsys Inc., "PrimeTime: Static timing analysis," 2024.
- [6] Synopsys Inc., "PrimePower: RTL to signoff power analysis," 2024.
- [7] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *CoRR*, vol. abs/1312.6203, 2013.
- [8] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Neural Information Processing Systems*, 2016.
- [9] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
- [10] Y. Zhang, H. Ren, and B. Khailany, "Grannite: Graph neural network inference for transferable power estimation," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.
- [11] Y. Du, Z. Guo, X. Jiang, Z. Chai, Y. Zhao, Y. Lin, R. Wang, and R. Huang, "Powpredict: Cross-stage power prediction with circuit-transformation-aware learning," in *Proceedings of the 61st ACM/IEEE Design Automation Conference, DAC '24*, (New York, NY, USA), Association for Computing Machinery, 2024.
- [12] Synopsys Inc., "Design compiler: Timing, area, power, & test optimization," 2024.
- [13] Synopsys Inc., "VCS," 2024.
- [14] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the iscas-85 benchmarks: A case study in reverse engineering," *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 72–80, 1999.
- [15] L. Amaru, P.-E. Gaillardon, E. Testa, and G. D. Micheli, "The EPFL combinational benchmark suite," Feb. 2019.
- [16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.