Bridging the Power Estimation Gap: A GNN-Based Prediction Model for Approximate Logic Synthesis

Fuxuan Li*, Ao Liu*, Siting Liu†, Hui Wang*, Jie Han‡, Honglan Jiang*
*School of Integrated Circuits, Shanghai Jiao Tong University, Shanghai, China
†School of Information Science and Technology, ShanghaiTech University, Shanghai, China
‡Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada
{fxli10, 2017la}@sjtu.edu.cn, liust@shanghaitech.edu.cn, ihuiwang@sjtu.edu.cn, jhan8@ualberta.ca, honglan@sjtu.edu.cn

***Abstract*—Approximate computing is an application-related paradigm that trades limited accuracy for improvements in hardware cost. As a key technique of approximate computing, approximate logic synthesis (ALS) automatically generates approximate circuits with reduced area, power, and delay while satisfying predefined quality-of-result (QoR) constraints. However, in typical gate-level ALS workflows, synthesis tools are invoked at the final stage for optimization, leading to a discrepancy between the circuit in design space exploration (DSE) and the final obtained circuit. Thus, the power estimation for a candidate circuit during DSE may exhibit a significant gap from the actual power consumed by its post-synthesis circuit. This gap may mislead the DSE to a sub-optimal design. To address this issue, we propose a graph neural network (GNN)-based power prediction model that operates on gate-level circuits. The model incorporates multi-head channel attention, which extracts high-level topological and functional features that correlate with power dissipation and implicitly captures the optimization behavior of synthesis tools. Thus, it enables a direct prediction of post-synthesis power from pre-synthesis gate-level circuits. Experimental results show that the proposed model improves the concordance index (C-index) for power ranking by up to 14.0% over traditional methods. Furthermore, we construct an ALS framework by integrating the proposed model with Cartesian genetic programming (CGP). Compared to state-of-the-art ALS approaches, our GNN-CGP framework generates circuits with up to 26.8% power savings under the same error constraints.**

***Index Terms*—approximate computing, approximate logic synthesis, graph neural network, Cartesian genetic programming**

I. Introduction

Recent advances in compute-intensive applications like artificial intelligence (AI), big data, and Internet of Things (IoT) have raised stringent demands on the performance and energy efficiency of computing hardware. However, as Moore's law slows down, traditional exact computing hardware struggles to meet these needs [1]. In response, approximate computing has emerged as a hardware-efficient paradigm that sacrifices limited accuracy for significant improvements in power, area, and speed [2], gaining attention in error-resilient areas such as image processing and machine learning [3].

In approximate computing, approximate logic synthesis (ALS) plays a key role, which focuses on automatically generating approximate circuits with reduced hardware cost while satisfying predefined quality-of-result (QoR) constraints [4]. To enable an effective design space exploration (DSE) for ALS, fast and precise error and hardware cost evaluations are of great importance. While efficient error modeling and QoR evaluation approaches [5]–[7] have been proposed, hardware cost evaluation still shows significant deficiencies.

Generally, the area estimation in DSE typically relies on the logic gate counts and cell parameters of candidate circuits [7]–[9]. Power estimation is commonly based on empirical formulas [10], which often deviate significantly from the actual values. Whether area-driven or power-driven, the circuits selected by the DSE process are ultimately subject to optimization by synthesis tools, that may lead to changes in the circuit implementation. This inconsistency leads to deviations between hardware estimates during exploration and the actual post-synthesis results. The deviations may impose a directional bias on the selection across generations, causing the DSE to drift and potentially overlook promising candidate circuits. Meanwhile, some ALS frameworks resort to traditional synthesis tools, e.g., Synopsys Design Compiler (DC), Yosys-abc [11], [12], to fully synthesize and map each candidate circuit, thereby providing accurate evaluations of area, power, and timing [13]. However, the high computational cost of these tools makes them unsuitable for large-scale DSE. This challenge becomes particularly critical in gate-level ALS, where changes are directly applied to gate-level circuits, generating a vast number of candidate circuit designs. In this case, the limitations of traditional evaluation methods in terms of speed and accuracy are further amplified. An effective framework capable of both accurately capturing gate-level structural features and efficiently predicting the hardware costs is still lacking.

As power dissipation is particularly challenging and critical in approximate computing, we propose a graph neural network (GNN)-based power prediction model specifically designed for gate-level ALS. The core idea is to treat the pre-synthesis gate-level circuits as a graph, with logic gates as nodes and interconnects as edges. Trained on this graph input, the GNN automatically extracts high-level topological and functional features that correlate with power and implicitly captures the optimization behavior of synthesis tools. Consequently, without time-consuming synthesis, the model can accurately predict the post-synthesis power directly from the corresponding pre-synthesis gate-level circuit. This capability explicitly bridges

This work was supported in part by the National Natural Science Foundation of China under Grant 62374108 and Grant 62204155, and in part by the Natural Sciences and Engineering Research Council of Canada under Grant RES0048688, Grant RES0051374, and Grant RES0054326.

the gap between the pre-synthesis gate-level circuits used during DSE and the post-synthesis optimized circuits at the end of the flow, enabling fast and reliable assessment of large candidate sets and guiding DSE more effectively.

To verify the effectiveness of the proposed model, we construct a global ALS framework by integrating the model with Cartesian genetic programming (CGP). In this framework, the prediction model enhances the accuracy of the power estimation, while CGP provides evolutionary optimization to systematically explore the design space and progressively approximate the Pareto front.

Experiments on benchmark circuits show that the proposed GNN-based power prediction model achieves high prediction accuracy and strong transferability across diverse gate-level circuit inputs. When combined with CGP, the ALS framework delivers a favorable balance of prediction accuracy and search efficiency. Under the same error constraints, it achieves superior Pareto fronts compared to representative ALS methods.

The main contributions are summarized as follows.

- A GNN-based prediction model for gate-level circuits is proposed, enabling a fast and accurate power prediction without invoking traditional synthesis tools.
- A global ALS framework is constructed by integrating the proposed GNN-based prediction model with CGP, resulting in an enhanced exploration of the design space.
- The proposed framework is validated on multiple benchmark circuits, demonstrating superior Pareto fronts compared with representative ALS method.

II. Related Work

A. Approximate Logic Synthesis Strategies

ALS can be broadly categorized into structural netlist transformation, Boolean rewriting, approximate high-level synthesis (AHLS), and evolutionary synthesis [2]. Structural netlist transformation reduces circuit complexity through gate-level pruning and rewiring [9], whereas Boolean rewriting does it through Boolean simplification [5]. AHLS, exemplified by [13], introduces approximation at high abstraction levels, enabling flexible accelerator design but with limited gate-level controllability. Evolutionary synthesis employs methods such as genetic algorithms and reinforcement learning to optimize digital circuits. EvoApprox8b [14] employs the CGP algorithm to explore the design space at the global level and constructs a diverse library of approximate arithmetic units.

Evolutionary synthesis generates numerous candidate designs, making efficient evaluation for DSE a central challenge. GNNs are promising for evolutionary synthesis due to their structural awareness and transferability. Prior studies show that GNNs can rapidly identify high-quality solutions from a vast pool of candidates [15], highlighting their potential to complement the exploratory capability of evolutionary algorithms and enhance the efficiency of multi-objective optimization.

B. Hardware Evaluation in ALS

In ALS, a fast and accurate evaluation for candidate circuits is crucial to guide the process to an optimal design. Thus, a significant amount of research has been conducted in the evaluation of circuits generated during ALS. As an early foundational study, ABACUS relies on Synopsys DC for accurate performance, power, and area (PPA) assessments in accelerator design [13]. However, such dependence on EDA tools incurred prohibitive runtime overhead, significantly limiting the scalability of DSE. To address this issue, AutoAX integrates machine learning (ML) models to accelerate the evaluation process, offering faster approximate PPA and QoR estimations [16]. Nevertheless, the simple ML models lack structure-awareness, making it difficult to capture circuit-level dependencies. As a result, prediction errors could grow substantially in complex designs, which in turn misled DSE.

In general, combinational circuits are represented as directed acyclic graphs (DAGs) that align naturally with GNNs. ApproxPilot adopts a two-layer GNN, where one layer captures path-criticality information and the other performs the regression for PPA and QoR [15]. By leveraging the structural features, ApproxPilot achieves more accurate evaluations, particularly in delay compared to conventional ML-based approaches. Furthermore, ApproxGNN combines embedding and prediction modules into a two-stage framework, which enhances model transferability and enables accurate prediction for accelerator performance with minimal fine-tuning [17]. Compared to traditional methods, GNN-based prediction models strike a balance between accuracy and speed in complex DSE.

Despite these advances, most existing methods focus on accelerator-level evaluations, which aim to choose proper approximate arithmetic units and accelerator architectures, resulting in limited generalization. At the finer gate-level, circuits exhibit richer structural details and more complex dependencies, which makes accurate and efficient evaluation an open challenge. Yet, these gate-level circuits also share inherent structural regularities, providing a foundation for evaluation models to achieve generalization across diverse designs.

III. Proposed Methodology

To bridge the power estimation gap, we propose a GNN-based prediction model for gate-level circuits and integrate it with CGP to enable gate-level ALS.

A. Framework

An overview of the proposed ALS framework is shown in Fig. 1, where the dataset construction and GNN training are also included. We construct a large, design-agnostic Register-Transfer Level (RTL) code dataset using a stochastic generator, synthesize each design to a gate-level netlist, and obtain the power labels under given test vectors. We then count the commonly used gate types in the netlists and perform gate-limited synthesis. The netlists are then preprocessed for the conversion from netlists to graphs. Pairing each graph with its power label yields a fully supervised dataset (Section III-B). Based on this dataset, the proposed GNN architecture for gate-level circuits modeling is trained (Section III-C).

For ALS, the target RTL is first synthesized with gate-type limitation and then encoded into an initial CGP genome that can mutate into approximate candidates. For each candidate,

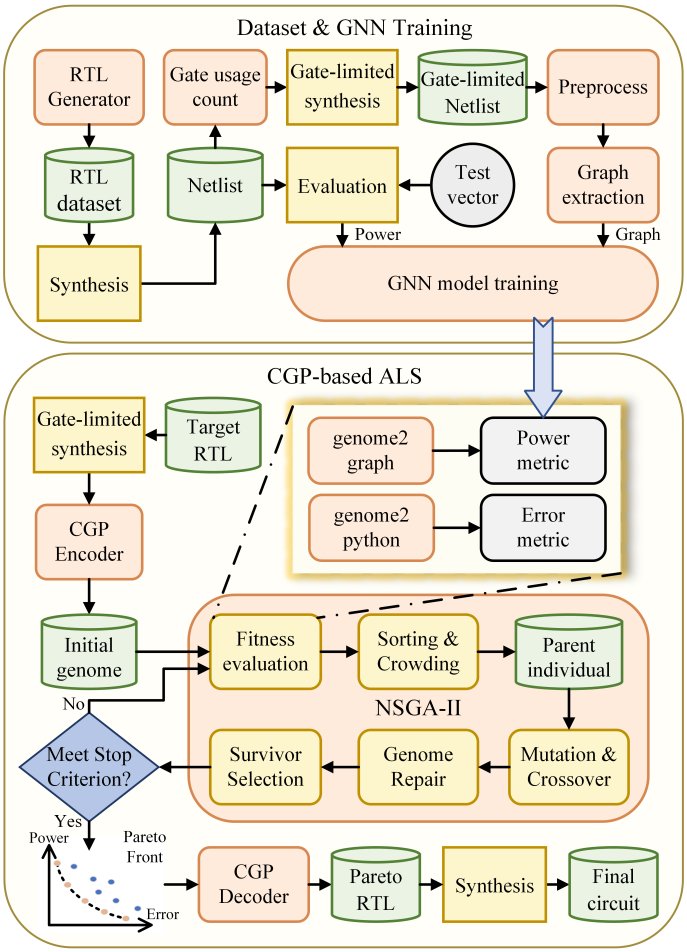


Fig. 1. Proposed GNN-CGP framework for approximate logic synthesis.

genome2graph converts the genome into a gate-level graph fed to the GNN to produce the power metric, while genome2python runs a CuPy-accelerated evaluator to compute the error metric under the given test vectors. The non-dominated sorting genetic algorithm II (NSGA-II) is employed to perform the bi-objective optimization [18]. When the stopping criterion is satisfied, the selected genomes are decoded into RTL and synthesized by the synthesis tool to obtain the final circuits.

B. Dataset Construction

To train the GNN model, we programmatically generate a large combinational RTL corpus. For each circuit, the bit widths for primary input (PI) A, primary output (PO) O, and intermediate signal D are sampled from $[8, 32]$, $[4, 32]$, and $[1, 12]$, respectively. D and O are populated bit by bit or in slices using random expressions of bitwise (shift, AND, OR, XOR) and arithmetic (add, subtract, multiply) operators with optional unary negation, and occasional constants 0 and 1. In this process, operands for D drawn from A and prior D bits, while operands for O drawn from A, D, and prior O bits. To ensure structural diversity and avoid operand repetition within a single expression, the operands are sampled with an anti-reuse weight $w(x) = 1/(1 + \text{count}(x))^2$. In addition, we include adders, subtractors, and multipliers at multiple bit widths to better cover arithmetic patterns.

As power consumption is a critical optimization target of approximate circuits, power is set as the supervision target for training the GNN. Each RTL code is synthesized with the Synopsys DC, resulting in a gate-level netlist that is used to generate power labels evaluated by Synopsys PrimeTime PX and VCS under uniformly distributed random test vectors. To standardize node types, we map all designs to a library of 14 frequently used and basic logic gates, i.e., `PULL_UP/DOWN`, `INV`, `AND/OR/XOR/NAND/NOR/XNOR_2`, `NAND/NOR_3`, `AOI/OAI_2_1`, `FULL_ADDER`. This process ensures graph consistency and mimics synthesis-induced structural deviations.

Before graph extraction, the obtained netlists are preprocessed to standardize the structure and apply data augmentation that mimics the gap introduced by synthesis. First, the topological re-indexing is performed to enforce a valid topological ordering of the nodes. Second, we deliberately inject edits that are typically optimized away during synthesis, including decomposition of complex cells into simpler gates, addition of redundant logic that does not affect PO, and local rewrites that preserve functional equivalence. These preprocessing steps effectively mimic the gap between the circuits explored during DSE and the final post-synthesis optimized circuits.

From each preprocessed netlist, we construct a gate-level DAG, where each gate instance is a node, edges connect a driver's output to the input pins of its fanout gates, and PI and PO are represented as special nodes. As the full adder has two distinct outputs, it is modeled by two nodes to separately represent SUM and COUT. The node feature is denoted by an 18-dimensional, technology-agnostic one-hot vector, i.e., 15 gate tokens consisting of 13 gate types plus `FULL_ADDER_SUM/COUT` and 3 special tokens `PI`, `PO`, and `NONE`. Edges are directed and untyped. The resulting graphs, paired with the power labels reported by Synopsys PrimeTime PX, form the supervised dataset.

C. GNN Model for Gate-Level Circuits

Fig. 2 shows the structure of the proposed GNN-based prediction model. To enable effective learning by the GNN, we propose a three-layer GNN model with the graph isomorphism network (GIN) [19] as the baseline. In addition, we incorporate gated residuals and multi-head channel attention (MHCA) to better capture higher-order interactions within the graph and the power dependencies between nodes.

In each GIN layer, the message passing and aggregation are performed to update the node features. Specifically, the update for layer k is calculated as

$$\tilde{h}_v^{(k)} = \text{MLP}_k\left((1+\varepsilon^{(k)})h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right), \quad (1)$$

where $h_v^{(k)}$ is the feature vector of node v at layer k, $\varepsilon^{(k)}$ is a learnable parameter, $\mathcal{N}(v)$ represents the in-neighbors of node v, and MLP_k is a two-layer perceptron with BatchNorm and ReLU operations. The reason for selecting GIN is that its aggregation method is based on summation that aligns well with how power is influenced by the cumulative effects of neighboring nodes and their switching behaviors.

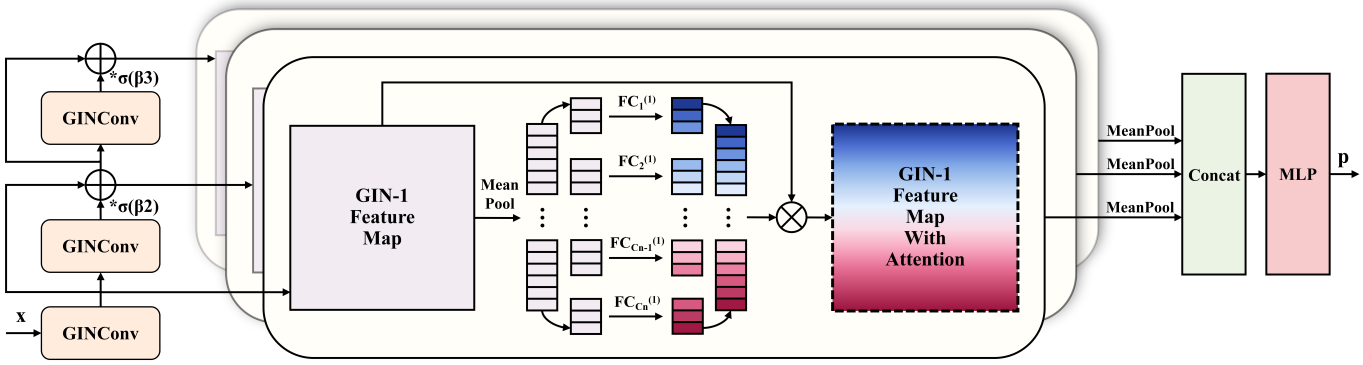


Fig. 2. Proposed GNN model structure.

To mitigate the typical over-smoothing problem in GNNs, we introduce gated residuals at each layer by

$$h_v^{(1)} = \tilde{h}_v^{(1)}, \tag{2}$$

$$h_v^{(2)} = h_v^{(1)} + \sigma(\beta_2)\,\tilde{h}_v^{(2)}, \tag{3}$$

$$h_v^{(3)} = h_v^{(2)} + \sigma(\beta_3)\,\tilde{h}_v^{(3)}, \tag{4}$$

where σ denotes the sigmoid function, β_2 and β_3 are learnable parameters. This gating mechanism regulates the effective depth of the residual connections, enabling robust long-range interaction modeling without over-smoothing.

For graph-level readout, we first compute the multi-scale summaries by mean pooling at each layer, i.e.,

$$g^{(k)} = \text{MeanPool}\left(\{h_v^{(k)} : v \in \mathcal{V}\}\right) \in \mathbb{R}^C, \quad k = 1, 2, 3. \tag{5}$$

By using mean pooling instead of the original add pooling, the model achieves better transferability across circuits of different sizes. We then apply the MHCA mechanism to each layer's node features. Specifically, for each $g^{(k)}$, we split it into C_n heads, each with a width of $d = \frac{C}{C_n}$. For head $i \in \{1, \ldots, C_n\}$,

$$s_i^{(k)} = \sigma\left(W_{i,2}^{(k)}\,\phi(W_{i,1}^{(k)}\,g_i^{(k)})\right) \in \mathbb{R}^d, \tag{6}$$

where $W_{i,1}^{(k)} \in \mathbb{R}^{m\times d}$, $W_{i,2}^{(k)} \in \mathbb{R}^{d\times m}$, ϕ is ReLU, and σ is a sigmoid function. After obtaining all head outputs, they are concatenated and then applied to the node features with a residual connection

$$s^{(k)} = \left[s_1^{(k)} \| \cdots \| s_{C_n}^{(k)}\right] \in \mathbb{R}^C, \; y^{(k)} = \left(s^{(k)} \odot h^{(k)}\right) + h^{(k)}, \tag{7}$$

where $h^{(k)} \in \mathbb{R}^{|\mathcal{V}|\times C}$, and the element-wise multiplication $\odot$ is applied along the channel dimension with broadcasting over all nodes. The channel attention mechanism can identify recurring cell types and structures in circuits, improving the model's sensitivity to power prediction. Finally, we use $y^{(k)}$ to compute $m^{(k)}$ for each layer based on (5). The outputs of all layers are then concatenated and passed through a simple two-layer MLP head to predict the power by

$$p = W_2\,\phi(W_1\left[m^{(1)} \| m^{(2)} \| m^{(3)}\right]) \in \mathbb{R}. \tag{8}$$

D. Design Space Exploration

In the proposed ALS framework, the DSE is performed based on CGP and NSGA-II. Fig. 3 shows the encoding scheme that encodes the gate-level circuit into a genome. A node is encoded by specifying (i) the gate type (underlined in the genome), and (ii) the indices of its input connections that can be either PI or the outputs of previous nodes. Each node generally uses one functional index, whereas a full adder (FA) is assigned with two output indices as it has two different outputs. For FA nodes, the parity of the index determines whether it refers to the sum or the cout. In addition, we randomly insert NONE-type nodes that enable neutral genetic drift and provide structural slack during mutation and crossover. Finally, the last segment of the genome specifies which nodes drive the PO.

In each generation, every individual is evaluated along two branches. First, the genome is unfolded into a gate-level DAG, where each node is assigned an 18-dimensional one-hot feature, and the graph is fed to the GNN model for power estimation. Second, the genome is converted into a CuPy vectorized evaluator, i.e., the fixed test vectors are moved to a GPU, each logic gate is mapped to a bitwise GPU function, and all node outputs are computed in parallel via a single topological pass, avoiding loops over input samples. The parent selection uses binary tournaments with nondominated rank and crowding distance; crossover and mutation are applied in the circuit-feasible space, with repair to ensure valid PO. The elitist selection is then performed by using NSGA-II on the combined parent and offspring populations to form the next generation. When a stop criterion is met, each Pareto-front genome is decoded back to a gate-level circuit that is synthesized for optimization.

IV. Experimental Results

A. Experiment Setup

The proposed GNN model is implemented in PyTorch Geometric. The circuit designs are implemented in Verilog HDL for synthesis and power evaluation. The designs are synthesized with Synopsys DC targeting a 28nm CMOS standardcell

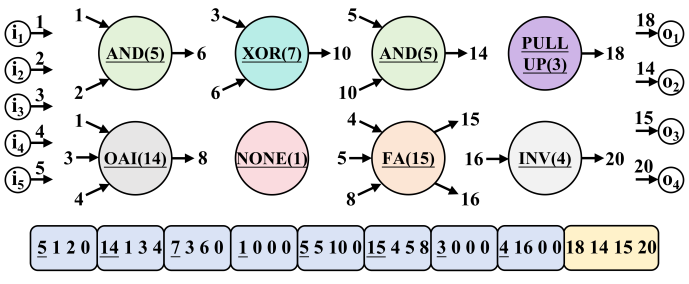


Fig. 3. Genome encoding in Cartesian genetic programming for circuits.

library. The power is evaluated by using Synopsys PrimeTime PX and VCS that captures the signal switching patterns with uniformly random test vectors. Following Section III-B, we generate the RTL codes for 10,000 combinational circuits. The 70/20/10% of the dataset are randomly sampled for training, validating, and testing the model. The training set is further augmented with 72 arithmetic circuits in Synopsys DesignWare library including adders, subtractors, and multipliers, spanning multiple bit widths in both signed and unsigned variants.

In DSE, the CGP genome length is capped at 2000 nodes to cover typical approximate circuit scales. The mean squared error (MSE) and mean relative error (MRE) are used as the error metrics to assess robustness under different criteria. We use EvoApproxLib (v2022) [20] as the primary baseline. EvoApproxLib (referred to as EAL in this work) is a state-of-the-art approximate arithmetic circuit library generated by using a CGP-based ALS framework, which enables a direct and fair assessment of our GNN-based prediction model. The widely used 8-bit signed and unsigned multipliers are selected from the EAL as benchmark circuits. The error budgets are bounded by MSE and MRE. The MSE with an upper limit of 3×10^5 is set for unsigned multipliers, whereas the MRE with an upper limit of 10% is used for signed multipliers. The initial population of the DSE is a combination of the exact circuits and their corresponding approximate circuits from EAL. NSGA-II runs for 10,000 generations, after which the non-dominated individuals are synthesized under identical optimization constraints to obtain the final Pareto set.

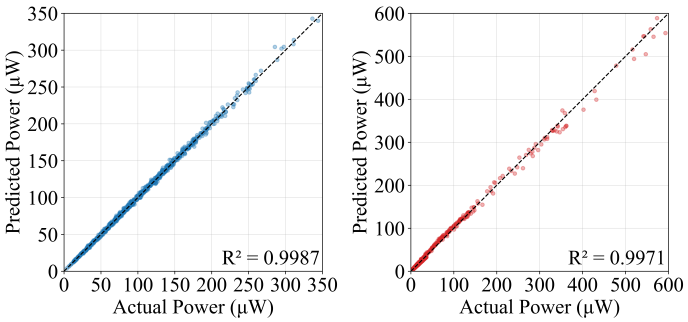
B. Power Prediction

Fig. 4(a) shows the power prediction results of the proposed model on the test set, and Fig. 4(b) shows the performance on the unseen EAL dataset without fine-tuning. The test set contains 1,000 circuits; the EAL set includes all designs except for the 15 designs exceeding the genome length limit, yielding 535 instances. In both cases, obtained prediction points align closely with the ideal $y{=}x$ line, resulting in high coefficients of determination. This indicate that the proposed GNN model learns the representations encoding high-level structural and functional characteristics related to power dissipation and implicitly captures the optimization behavior of synthesis tools. Notably, although the power consumption results of the designs in EAL are beyond the range covered by our training and test data, the model maintains an excellent fit, demonstrating strong transferability under distribution shift and an ability to extrapolate. These experimental results confirm that the proposed GNN-based prediction model delivers accurate estimates for seen circuits and generalizes effectively to unseen circuits, making it a robust evaluator for ALS.

Moreover, we compare the proposed model against representative GNN baselines, GCN, GAT, and GIN, and an empirical formula. The empirical formula adopts an analytic switching-activity model utilized in [20]. The total power of a circuit is defined as $P = P_s + P_d$, where P_s is the sum of cell leakages and P_d represents the dynamic power. P_d is calculated as $P_d = 0.5\,\alpha\,C_{\text{load}}V_{dd}^2 f$, where C_{load} is the sum of the input-pin capacitances of fanout, V_{dd} is the supply voltage, f is the clock frequency, and α is the activity factor obtained from simulation. We report the coefficient of determination (R^2) and a ranking fidelity measured by the concordance index (C-index). As the empirical formula shows substantial mismatch relative to ground truth, we normalize its predictions before computing R^2 to avoid trivially deflating the score. This normalization is monotonic and thus does not affect the C-index. Considering that newly generated individuals in evolution are often similar, we compute the C-index on near-tie pairs only. We collect all unordered pairs (i, j) whose ground-truth powers differ by less than a threshold τ and uniformly sample $N_{\text{pairs}} = 10^5$ pairs for evaluation, where $\tau = 0.05 \cdot \max p_i$. The C-index is given by

$$\text{C-index} = \frac{1}{N_{\text{pairs}}} \sum_{(i,j)\in\mathcal{S}_\tau} \mathbf{1}\{(p_i - p_j)(\hat{p}_i - \hat{p}_j) > 0\}\,. \quad (9)$$

where $\mathcal{S}_\tau$ denotes the sampled near-tie pair set and $\mathbf{1}\{\cdot\}$ is the indicator function.



(a) Test dataset (b) EvoApproxLib dataset

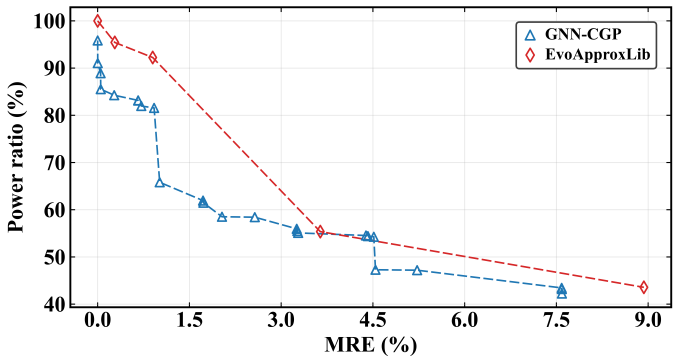
Fig. 4. Predicted versus actual power across datasets: (a) Test dataset; (b) Transfer evaluation on the unseen EvoApproxLib dataset without fine-tuning.

TABLE I
COMPARISON OF MODEL PERFORMANCE ACROSS DATASETS

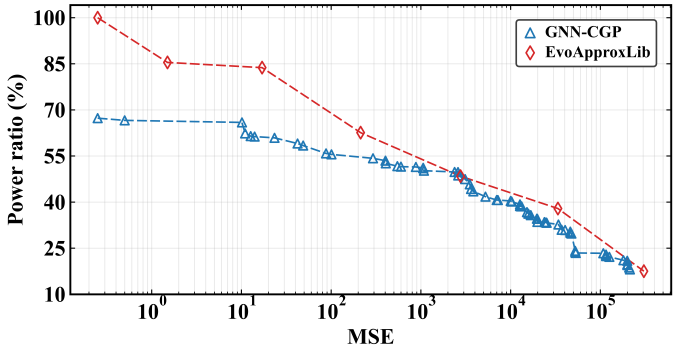
	Test dataset		EvoApproxLib		RTL-OPT[a]	
Model	**R^2**	**C-index**	**R^2**	**C-index**	**R^2**	**C-index**
Formula	0.9568	81.35%	0.9665	86.36%	0.9138	62.30%
GCN	0.9953	90.66%	0.9928	92.55%	0.9836	81.25%
GAT	0.9827	88.23%	0.9019	91.17%	0.9435	83.90%
GIN	0.9956	91.93%	0.9923	93.43%	0.9775	81.10%
Proposed	**0.9987**	**95.39%**	**0.9971**	**95.75%**	**0.9852**	**86.57%**

[a] 16 combinational designs that satisfy the experimental setups of the proposed model in RTL-OPT are tested.

As shown in Table I, all GNN-based models substantially outperform the empirical formula on the test set and they exhibit strong transferability to EAL except GAT. Among the GNN-based models, GIN surpasses GCN and GAT, consistent with the advantage of the sum aggregation for modeling the additive contributions to power. Our model further achieves the highest R^2 and C-index on both the test and EAL sets with R^2=0.9987 and 0.9971 and C-index=95.39% and 95.75%, respectively. It improves the C-index for power ranking by up to 14.0% over the empirical formula on the test set and by 9.4% on EAL, thereby directly benefiting the DSE. In addition, to evaluate the efficiency of the models on various types of datapaths, they are tested on 16 circuit designs from the RTL-OPT dataset [21]. Note that 16 designs satisfying the experimental setup requirements of the models are selected from RTL-OPT, including alu, comparator, divider, adder, subtractor,



(a) 8-bit signed multiplier



(b) 8-bit unsigned multiplier

Fig. 5. Pareto fronts for approximate multipliers. (a) shows the power vs. MRE results for 8-bit signed multipliers with MRE on linear scale, while (b) presents the power vs. MSE results for 8-bit unsigned multipliers with MSE on logarithmic scale. Both demonstrate that our GNN-CGP framework (blue triangles) achieves better power-error trade-offs compared to EvoApproxLib baselines (red diamonds).

mul_const, mul_subexpression, mult_if, mux_dead of different widths. Table I shows that the proposed model also achieves the best performance. The observed degradation in C-index across models can be attributed to the limited number of samples and the small differences among power values.

C. Approximate Logic Synthesis

Fig. 5 compares the Pareto fronts obtained by the proposed ALS framework based on GNN-CGP against those from EAL. The GNN-CGP curve lies significantly below that of EAL across almost the entire considered error range. This advantage is particularly pronounced in the high-accuracy regime, where circuit structures tend to be more complex, demonstrating that the proposed prediction model can more effectively guide the evolutionary search toward high-quality solutions. Quantitative analysis shows that under identical error constraints, GNN-CGP framework achieves up to 26.8% power savings compared to the EAL baseline. Moreover, the GNN model contains only about 65k parameters and achieves an inference time of about 0.3 ms per sample, which is negligible in runtime compared with error computation. In contrast, traditional power estimation for the considered size of design using commercial EDA tools typically require seconds to minutes. As a result, the proposed model can be integrated into the DSE in ALS as a near-zero-cost enhancement.

D. Ablation Experiment

To verify that improper estimators can misguide DSE, we conduct an ablation experiment comparing the empirical formula against the proposed GNN-based prediction model under identical conditions. The error bounds are selected from the MSE values of the initial population. The resulting power savings and actual MSE values are reported in Table II. Note that the power saving for each method is measured relative to the initial circuit at the corresponding error bound shown in the first column, rather than against the exact circuit.

A power saving of 0% indicates a failure to escape the local optimum, while a negative value suggests that the initial population that should have remained on the Pareto front is erroneously excluded. Across all tested error bounds, the proposed model exhibits a consistent performance advantage over

TABLE II
ABLATION RESULTS UNDER DIFFERENT ERROR BOUNDS

Error bound (MSE)	Empirical formula		GNN prediction model	
	Power saving	MSE	Power saving	MSE
213	0%	213	**+11.29%**	**101**
2764	-2.15%	2370	-0.46%	2608
33655	-27.47%	27184	**+12.23%**	**25219**

the empirical formula. This suggests that the DSE using the empirical formula struggles to achieve meaningful optimization from locally optimal initial populations. This is because the empirical approach tends to overlook certain promising intermediate candidate designs whose structures appear complex superficially yet can be effectively optimized by synthesis tools, thereby misguiding the search trajectory. In contrast, the GNN-based prediction model captures the optimization behavior of synthesis tools, retains those seemingly complex novel candidate, and maintains solution diversity, ultimately enabling more effective optimization through DSE.

V. CONCLUSION

This paper aims to bridge the power estimation gap that may misguide the gate-level ALS. We propose a GNN-based prediction model that operates directly on pre-synthesis gate-level circuits and outputs post-synthesis power, reducing the reliance on time-consuming EDA tools. Specifically, the model builds on a three-layer GIN with gated residuals and multi-head channel attention. It is trained on a carefully constructed dataset without pronounced structural bias, which is particularly processed to mimic synthesis-induced structural changes. The proposed model attains high R^2 and ranking fidelity, improving the C-index for near-tie power ranking by up to 14.0% over traditional methods and exhibiting strong transferability to unseen circuits.

We further integrated the prediction model into a CGP-driven ALS framework, which generates superior power-error Pareto fronts. Under the same error constraints, these fronts achieve up to 26.8% power savings compared to representative ALS baselines. By capturing structural features and synthesis-tool optimizations, our GNN-based prediction model effectively bridges the evaluation gap, leading to a more reliable and scalable design space exploration.

REFERENCES

[1] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[2] A. M. Dalloo, A. J. Humaidi, A. K. Al Mhdawi, and H. Al-Raweshidy, "Approximate computing: Concepts, architectures, challenges, applications, and future directions," *IEEE Access*, 2024.

[3] V. Leon, M. A. Hanif, G. Armeniakos, X. Jiao, M. Shafique, K. Pekmestzi, and D. Soudris, "Approximate computing survey, Part II: Application-specific & architectural approximation techniques and applications," *ACM Computing Surveys*, vol. 57, no. 7, pp. 1–36, 2025.

[4] I. Scarabottolo, G. Ansaloni, G. A. Constantinides, L. Pozzi, and S. Reda, "Approximate logic synthesis: A survey," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2195–2213, 2020.

[5] S. Hashemi, H. Tann, and S. Reda, "BLASYS: Approximate logic synthesis using Boolean matrix factorization," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[6] X. Wang, X. Zhou, S. Han, R. Dai, X. Shen, M. Xu, L. Ni, W. Wu, and W. Qian, "AccALS 2.0: Accelerating Approximate Logic Synthesis by Simultaneous Selection of Multiple Local Approximate Changes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[7] C. Meng, A. Mishchenko, W. Qian, and G. De Micheli, "Efficient Resubstitution-Based Approximate Logic Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[8] C.-T. Lee, Y.-T. Li, Y.-C. Chen, and C.-Y. Wang, "Approximate logic synthesis by genetic algorithm with an error rate guarantee," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 2023, pp. 146–151.

[9] L. Witschen, T. Wiersema, M. Artmann, and M. Platzner, "MUSCAT: MUS-based circuit approximation technique," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 172–177.

[10] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*. IEEE, 2016, pp. 1–6.

[11] C. Wolf, J. Glaser, and J. Kepler, "Yosys-a free Verilog synthesis suite," in *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, vol. 97, 2013.

[12] A. Mishchenko *et al.*, "ABC: A system for sequential synthesis and verification," *URL http://www.eecs.berkeley.edu/alanmi/abc*, vol. 17, 2007.

[13] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "ABACUS: A technique for automated behavioral synthesis of approximate computing circuits," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.

[14] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 258–261.

[15] Q. Zhang, S. Liu, Y. Hui, and C. Liu, "ApproxPilot: A GNN-based Accelerator Approximation Framework," in *2025 International Symposium of Electronics Design Automation (ISEDA)*. IEEE, 2025, pp. 226–231.

[16] V. Mrazek, M. A. Hanif, Z. Vasicek, L. Sekanina, and M. Shafique, "autoax: An automatic design space exploration and circuit building methodology utilizing libraries of approximate components," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[17] O. Vlcek and V. Mrazek, "ApproxGNN: A Pretrained GNN for Parameter Prediction in Design Space Exploration for Approximate Computing," *arXiv preprint arXiv:2507.16379*, 2025.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[20] V. Mrazek, Z. Vasicek, and L. Sekanina, "Evoapproxlib: Extended library of approximate arithmetic circuits," in *Proc. Workshop Open-Source EDA Technol.(WOSET)*, 2019, p. 10.

[21] Y. Lu, S. Liu, H. Zhou, W. Fang, Q. Zhang, and Z. Xie, "A New Benchmark for the Appropriate Evaluation of RTL Code Optimization," *arXiv preprint arXiv:2601.01765*, 2026.