

Cross-Design Generalization of Lightweight Surrogate Models for Predicting Logic Synthesis Quality

Junyan Huang*

School of Chemical Engineering and Light Industry, Guangdong University of Technology, Guangzhou, Guangdong, China

**Author to whom correspondence should be addressed.*

Copyright: © 2026 Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0), permitting distribution and reproduction in any medium, provided the original work is cited.

Abstract: The quality of results (QoR) of logic synthesis depends on the type and order of the optimization operators, and exhaustive evaluation is expensive, so machine-learning surrogates are widely used to predict recipe QoR. In the reference train and test lists released with the public dataset OpenABC-D, every design in the test list also appears in the training list, so figures reported on those lists need not reflect transfer to a new design. Using that dataset, this study takes 27 open-source circuits and 1,500 twenty-step ABC recipes, forming 40,500 labeled runs, trains four lightweight surrogates on 30 features, and scores them within each design under a random split and under leave-one-design-out (LODO). The median within-design mean absolute percentage error for area rises from 2.66% to 23.30% between the two protocols and falls back to 3.54% when the target is expressed relative to one reference-recipe calibration, although the coefficient of determination stays below zero. Under LODO, the median within-design rank correlation for area is 0.547, positive for 24 of the 27 held-out designs, so ranking information is partly transferable; for delay it is 0.007 ($P=0.097$) and no significant transfer is detected. Gradient boosting ranks best and trains in 0.161 seconds per fold. The evaluation protocol markedly changes apparent accuracy, and cross-design use of these features is supported for ranking area, not delay.

Keywords: Logic synthesis; Quality of results prediction; Surrogate model; Cross-design generalization; And-inverter graph

Online publication: October 21, 2026

1. Introduction

Logic synthesis converts a register-transfer level description into a gate-level netlist, and the quality of the result depends on the combination and order of optimization operators. In the ABC tool, operators such as rewrite, refactor, resub, and balance can be arranged freely, so the recipe space grows exponentially with

length, while one run of synthesis plus technology mapping takes considerable time, which makes exhaustive search infeasible ^[1].

Two lines of work address this. One searches the recipe space directly, approaching good recipes by reinforcement learning or multi-armed bandits ^[2–3]; the other trains surrogate models that replace real synthesis at very low inference cost and turn the search into a ranking of predicted values ^[4].

That value depends on the deployment scenario: the case with real engineering meaning is to apply a model trained on historical designs to one never seen before, avoiding resampling. Evaluation, however, is often organized by recipe. The reference train and test lists distributed with OpenABC-D are of this kind: all nine designs in the released test list also appear in the training list, so a model scored on those lists has already seen other recipes of every circuit. Under such a split, the model needs only to learn the design's scale plus the recipe's small perturbation, and the scale is precisely what is unknown across designs.

Using OpenABC-D, this study compares lightweight surrogates under the two protocols and answers three questions: how much the protocol changes apparent accuracy; how the target parameterization changes cross-design error; and whether ranking ability survives when absolute accuracy does not ^[5]. The contribution is to quantify these differences and show that area and delay differ in cross-design transfer.

2. Literature review

Machine-learning methods for logic synthesis fall into two lines. The search line begins with the synthesis flow developed without human knowledge ^[2]; DRiLLS models operator selection as reinforcement learning, FlowTune uses a domain-specific multi-armed bandit, and later work tunes synthesis-tool parameters jointly ^[3, 6–7]. Each step needs real synthesis feedback, so cost grows with the number of steps.

The prediction line avoids that cost. LOSTIN encodes the circuit with a graph neural network and the recipe with a recurrent network ^[4]; MTLSO attributes overfitting to the small number of available netlists and adds multi-task learning ^[8]. Others predict and-inverter graph (AIG) timing or power after approximate synthesis ^[9–10]. Circuit representation learning has advanced in parallel: DeepGate2 and DeepGate3 improve functional awareness and scalability, and PolarGate targets the functionality-representation bottleneck of graph neural networks on AIGs ^[11–13].

On data and evaluation, OpenABC-D labels 1,500 recipes on 29 open-source modules, and OpenLS-DGF offers a dataset-generation framework, while Chowdhury et al. argue the field lacks benchmarks comparable to the image domain ^[5, 14–15]. Generalization is studied in adjacent tasks: few-shot transfer to an unseen netlist, partitioning of unseen netlists, a generalizable surrogate for circuit topology design, flow runtime prediction, and tool-parameter rank prediction ^[16–20]. This study places the two protocols side by side under one dataset, feature set, and model family, and characterizes the transfer ceiling of lightweight features.

3. Methodology

3.1. Data and ground truth

The data are the machine-learning-ready release of OpenABC-D, which runs 1,500 recipes for each circuit ^[5]. Each recipe is a twenty-step sequence over seven operators: balance plus rewrite, refactor, and resub, each of the latter three in a plain and a zero-cost variant. The dataset records the final AIG's AND count, NOT count, and longest path length, together with post-mapping area and delay. This study keeps the 27 circuits with

both QoR labels and a published original netlist, yielding 40,500 observations. These circuits span a wide range: the original AND count lies between 403 and 114,771, the logic depth between 13 and 1,522, and the area differs by about 304 times across designs.

3.2. Lightweight features

The feature set is deliberately lightweight: 30 dimensions, with no graph-representation-learning module. The recipe side contributes 20 dimensions: the occurrence count of each of the seven operators, the normalized center of gravity of the positions at which each operator appears, the total counts of the four operator families, the number of zero-cost variants, and the number of operator switches between adjacent steps. The design side contributes 10 dimensions counted from the original netlist: counts of primary inputs, primary outputs, AND nodes, NOT nodes and total nodes, the logic depth, the mean and maximum fanout, the AND count per primary input, and the logic depth over the base-two logarithm of the AND count.

3.3. Target parameterization

The absolute target predicts the area or the delay after mapping directly. The relative target is defined as

$$\tilde{y}_{d,r} = \frac{y_{d,r}}{y_{d,r_0}} \quad (1)$$

where $y_{d,r}$ is the quality of results of design d under recipe r , and y_{d,r_0} is one reference recipe fixed across all designs. Every design runs through the same 1,500 recipes, so r_0 was fixed in advance as the first of that shared set, not chosen after inspecting results. Obtaining y_{d,r_0} for a new design costs one synthesis run; the reference sample is removed at scoring time so the model cannot inflate its score by reproducing the constant 1.

3.4. Evaluation protocols and metrics

The random split shuffles all samples into five-fold cross-validation, so other recipes of the same design appear in the training set. Leave-one-design-out tests on one circuit and trains on the remaining 26, giving 27 folds. Both protocols are scored within each design, against the dispersion of that design itself across recipes: the within-design coefficient of determination is one minus the ratio of summed squared prediction error to summed squared deviation from that design's recipe mean, so a model cannot score merely by guessing the design's overall scale. The study also reports the within-design mean absolute percentage error (MAPE), the within-design Spearman rank correlation, and a normalized top-1 regret: the selected recipe's true value minus the design minimum, divided by its true-value range, selection being the lowest prediction; 0 means the truly optimal recipe was selected and 1 the worst.

3.5. Models and statistical testing

Four fixed, untuned models are compared: standardized ridge regression (regularization strength 1.0); a 200-tree random forest (minimum leaf size 2); histogram-based gradient boosting (300 iterations, learning rate 0.08); and a ReLU multilayer perceptron with two 64-unit hidden layers, standardized inputs and targets, mean-squared-error loss and Adam (learning rate 0.001, batch size 1024) for 60 epochs. Unstated settings use library defaults; all models share the feature set and the seed 20260807, with scikit-learn 1.8 and PyTorch 2.11. Under the random split, each sample is scored from the fold in which it was held out, so scoring is entirely out-of-fold. The three directional comparisons, protocol, target parameterization, and rank correlation against

zero, use one-sided paired Wilcoxon signed-rank tests; the model-family comparison is two-sided. The 27 designs are the pairing units at a 0.05 significance level.

4. Results

Figure 1 shows the main results, with per-design area error on a logarithmic axis in panel (a).

Under the absolute target, the protocol difference is large. The median within-design coefficient of determination of gradient boosting falls from 0.141 under the random split to -51.2 under leave-one-design-out, and the median within-design MAPE rises from 2.66% to 23.30%, with a paired test of $P=7.5 \times 10^{-9}$. The coefficient of determination is strongly negative because its denominator is the dispersion of the design itself across recipes, whose median coefficient of variation is only 3.47%: a value of -51.2 means the squared error is about 52 times that within-design total, far worse than predicting the mean of the held-out design.

Switching to the relative target of Equation (1) removes most of that error. Under leave-one-design-out the median error of gradient boosting drops to 3.54% and the median coefficient of determination rises to -0.787 , with a paired test of $P=3.7 \times 10^{-8}$ for both improvements. The percentage error thus returns to the order seen under the random split, while the coefficient of determination stays below zero: the calibrated model still does not beat the mean of the held-out design. The shift is consistent with much of the cross-design error being the unknown overall scale, which one reference run measures.

Whether ranking ability transfers depends on the target. For area, the median within-design rank correlation under leave-one-design-out is 0.547 with an interquartile range of 0.150 to 0.699, and 24 of the 27 held-out designs take positive values, with $P=1.0 \times 10^{-6}$. For delay, by contrast, the median rank correlation is only 0.007 with an interquartile range of -0.024 to 0.090; 17 of the 27 designs take positive values, and $P=0.097$, which does not reach the significance level. **Figure 1(b)** shows this divergence directly, with the 27 designs ordered by their area rank correlation: the points for area lie clearly towards the positive side, whereas the points for delay are densely distributed near zero.

Selection quality follows the same pattern, more modestly. With the relative target under leave-one-design-out, the median top-1 regret for area is 0.234 (interquartile range 0.131 to 0.418) against 0.295 for the absolute target and 0.160 (interquartile range 0.085 to 0.287) under the random split. The regret of a recipe drawn uniformly at random has a median of 0.414 over the 27 designs, so the area surrogate improves on random selection without reliably recovering the best recipe. For delay, the median regret is 0.392 against a random reference of 0.358, so no improvement over random selection is observed.

The protocol gap holds for every model tested: the median area MAPE under leave-one-design-out is 26.42% for ridge regression, 21.29% for the random forest, 23.30% for gradient boosting and 24.92% for the multilayer perceptron, against 16.68%, 2.11%, 2.66% and 6.36% respectively under the random split. Gradient boosting ranks best, at 0.547 against 0.471 for the random forest, 0.455 for the multilayer perceptron and 0.389 for ridge regression; it beats the perceptron on 24 of the 27 designs, with $P=3.0 \times 10^{-5}$, and trains in 0.161 seconds per fold.

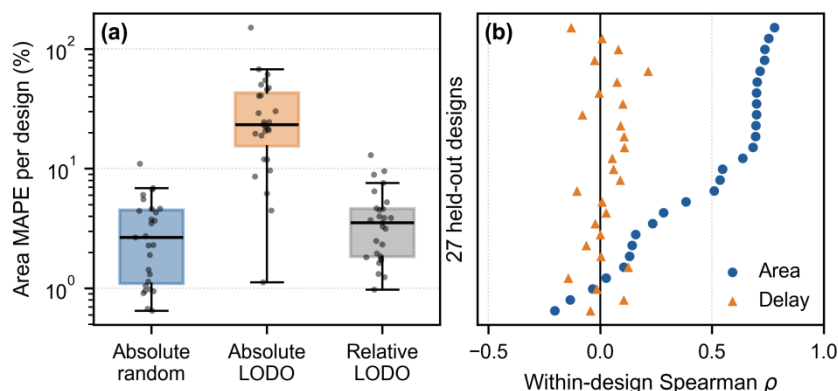


Figure 1. Cross-design generalisation of the gradient-boosted surrogate. (a) Area MAPE per design. (b) Within-design rank correlation under leave-one-design-out

5. Discussion

The apparent accuracy of a surrogate is highly sensitive to the evaluation protocol: an area error that looks ready for use under a random split grows by nearly an order of magnitude once the same model class and feature set are scored on an unseen design. Reports of such models should therefore state the splitting scheme explicitly.

The relative target points to a low-cost engineering path, since one synthesis run measures the unknown scale of a new design; what survives calibration is ordering rather than absolute accuracy.

The divergence between area and delay deserves attention. The median coefficient of variation of delay across recipes is 14.84%, higher than the 3.47% of area, so delay does not lack variation; rather, these features did not capture it, the median rank correlation being 0.007 with $P=0.097$. Delay is set by the critical path after technology mapping, whose link to the operator sequence is more indirect than that of area; the design tests whether transfer occurs, not why it fails. Cross-design transferability should therefore be verified separately for each target, and not extrapolated from the area result to delay.

This study has four boundaries. First, all evidence comes from OpenABC-D under one technology library and one recipe family, 1,500 twenty-step sequences over seven ABC operators, so applicability to other flows remains to be verified. Second, every design runs through that same shared recipe set, so the protocol measures transfer to an unseen design under already-represented recipes, not to unseen recipe sequences. Third, the effective sample size on the design axis is 27, not 40,500, and those circuits are not fully independent, since three are variants of the same AES core. Fourth, the feature set is lightweight by construction, so the observed ceiling reflects this feature class, and graph-representation models such as DeepGate3 or PolarGate may transfer better^[12–13].

6. Conclusion

This study compares lightweight surrogates under a random split and leave-one-design-out on the 27 OpenABC-D circuits and their 1,500 shared recipes. Three conclusions follow. First, the protocol markedly changes apparent accuracy: the median within-design area error rises from 2.66% to 23.30%, so figures from a recipe-level split cannot represent cross-design deployment. Second, a relative target calibrated by one

reference recipe reduces that error to 3.54% while the coefficient of determination stays negative, so the gain is scale calibration rather than prediction of recipe-to-recipe variation. Third, transfer of ranking depends on the target: the median rank correlation for area reaches 0.547, positive on 24 of the 27 designs, whereas for delay it is 0.007 with $P=0.097$ and no significant transfer is detected. Future work can test these conclusions on other synthesis flows and target delay-specific features.

Disclosure statement

The author declares no conflict of interest.

References

- [1] Brayton R, Mishchenko A, 2010, ABC: An Academic Industrial-strength Verification Tool, in Computer Aided Verification, Lecture Notes in Computer Science. Springer, Berlin, 24–40. https://doi.org/10.1007/978-3-642-14295-6_5
- [2] Yu C, Xiao H, De Micheli G, 2018, Developing Synthesis Flows without Human Knowledge. Proc. 55th Annual Design Automation Conference (DAC), 2018, 1–6. <https://doi.org/10.1145/3195970.3196026>
- [3] Hosny A, Hashemi S, Shalan M, et al., DRiLLS: Deep Reinforcement Learning for Logic Synthesis. Proc. 25th Asia and South Pacific Design Automation Conference (ASP-DAC), 2020, 581–586. <https://doi.org/10.1109/ASP-DAC47756.2020.9045559>
- [4] Wu N, Lee J, Xie Y, et al., 2022, LOSTIN: Logic Optimization via Spatio-Temporal Information with Hybrid Graph Models. Proc. IEEE 33rd Int. Conf. Application-specific Systems, Architectures and Processors (ASAP), 11–18. <https://doi.org/10.1109/ASAP54787.2022.00013>
- [5] Chowdhury AB, Tan B, Karri R, et al., 2021, OpenABC-D: A Large-Scale Dataset for Machine Learning Guided Integrated Circuit Synthesis. arXiv: 2110.11292. <https://doi.org/10.48550/arXiv.2110.11292>
- [6] Neto WL, Li Y, Gaillardon PE, et al., 2023, FlowTune: End-to-end Automatic Logic Optimization Exploration via Domain-Specific Multiarmed Bandit. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 42(6): 1912–1925. <https://doi.org/10.1109/TCAD.2022.3213611>
- [7] Cui Z, Shen M, 2024, Automatic Multi-parameter Tuning for Logic Synthesis with Reinforcement Learning. Proc. 2nd Int. Symp. Electronics Design Automation (ISED), 318–323. <https://doi.org/10.1109/ISED62518.2024.10617552>
- [8] Faez F, Karimi R, Zhang Y, et al., 2025, MTLSo: A Multi-task Learning Approach for Logic Synthesis Optimization. Proc. 30th Asia and South Pacific Design Automation Conference (ASP-DAC), 72–78. <https://doi.org/10.1145/3658617.3697721>
- [9] Jiang W, Yan J, Sapatnekar SS, 2025, ML-based AIG Timing Prediction to Enhance Logic Optimization. Proc. Design, Automation and Test in Europe Conference (DATE), 1–2. <https://doi.org/10.23919/DATE64628.2025.10993071>
- [10] Li F, Liu A, Liu S, et al., 2026, Bridging the Power Estimation Gap: A GNN-based Prediction Model for Approximate Logic Synthesis. Proc. Design, Automation and Test in Europe Conference (DATE), 1–7. <https://doi.org/10.23919/DATE69613.2026.11539351>
- [11] Shi Z, Pan H, Khan S, et al., 2023, DeepGate2: Functionality-aware Circuit Representation Learning. Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 1–9. <https://doi.org/10.1109/ICCAD57390.2023.10323798>

- [12] Shi Z, Zheng Z, Khan S, et al., 2024, DeepGate3: Towards Scalable Circuit Representation Learning. Proc. 43rd IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 1–9. <https://doi.org/10.1145/3676536.3676791>
- [13] Liu J, Zhai J, Zhao M, et al., 2024, PolarGate: Breaking the Functionality Representation Bottleneck of And-inverter Graph Neural Network. Proc. 43rd IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 1–9. <https://doi.org/10.1145/3676536.3676834>
- [14] Ni L, Wang R, Liu M, et al., 2025, OpenLS-DGF: An Adaptive Open-source Dataset Generation Framework for Machine-Learning Tasks in Logic Synthesis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 44(10): 3830–3843. <https://doi.org/10.1109/TCAD.2025.3555506>
- [15] Chowdhury AB, Thakur S, Pearce H, et al., 2023, Invited Paper: Towards the Imagenets of ML4EDA. Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 1–7. <https://doi.org/10.1109/ICCAD57390.2023.10323663>
- [16] Chowdhury B, Tan B, Carey R, et al., 2023, Bulls-Eye: Active Few-shot Learning Guided Logic Synthesis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42(8): 2580–2590. <https://doi.org/10.1109/TCAD.2022.3226668>
- [17] Yang J, Chen K, Qiao J, et al., 2026, Hypergraph Pseudo-Label Learning with Neighborhood Consistency for Unseen Netlist Partitioning. Computer-Aided Design, 2026(199): 104115. <https://doi.org/10.1016/j.cad.2026.104115>
- [18] Lu H, Fan S, Zhang S, et al., 2026, Towards Generalizable and Efficient Circuit Topology Design: A Graph-Transformer-Based Surrogate Model with Curriculum Learning. ACM Transactions on Design Automation of Electronic Systems, 31(4): 1–29. <https://doi.org/10.1145/3768167>
- [19] Tseng IL, 2024, Pragmatic EDA Flow Runtime Prediction with Machine Learning and Automatic Outlier Removal. Proc. IEEE Asia Pacific Conf. Circuits and Systems (APCCAS), 203–207. <https://doi.org/10.1109/APCCAS62602.2024.10808773>
- [20] Won D, Kim T, 2025, EDA Tool Parameter Optimization through Fast and Reliable Machine-Learning Based Rank Prediction Model. Proc. ACM/IEEE 7th Symp. Machine Learning for CAD (MLCAD), 1–7. <https://doi.org/10.1109/MLCAD65511.2025.11189232>

Publisher's note

Bio-Byword Scientific Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.