CELLBENCH: BENCHMARKING GENERALIZATION IN FAST STANDARD CELL EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Standard cell libraries are collections of reusable building blocks for very large scale integration (VLSI) circuits. Developing and optimizing cell libraries involves exploring a large design space of pre-layout netlists with diverse size configurations, topologies, and functions. For each candidate netlist, accurate evaluation of power, performance, and area (PPA) requires time-consuming layout generation and simulation, creating a bottleneck in design exploration. Machine learning can alleviate this bottleneck through fast prediction, but generalization to unseen cell designs remains insufficiently evaluated. To address this evaluation gap, we introduce **CellBench**, the first open dataset and benchmark for fast standard cell evaluation, covering PPA and constrained routability prediction and evaluating generalization to unseen size configurations, topologies, and functions. Building upon the ASAP7 7 nm FinFET process design kit, we generate diverse pre-layout netlists through cell library extension, topology generation, and transistor size assignment, yielding 118,906 netlists for PPA prediction and 296,274 for constrained routability prediction. Evaluation of existing prediction methods shows declining overall performance from unseen size configurations to unseen topologies and functions. Further analysis shows that structural similarity to training circuits helps explain PPA prediction errors, whereas functional similarity alone does not consistently reflect prediction difficulty. The benchmark is available for anonymous review.[1]

1 INTRODUCTION

As fundamental building blocks of modern integrated circuits, standard cells strongly influence circuit-level power, performance, and area (PPA), making cell library optimization important for achieving competitive chip performance. Figure 1 summarizes the standard cell library optimization flow. Standard cell netlist design, shown on the left, begins with selecting a set of commonly used logic functions. Library extension can further introduce new logic functions motivated by combinations of cells in existing designs (Liang et al., 2022; Fu et al., 2025). For each logic function, topology optimization seeks alternative transistor networks that implement the same logic with fewer transistors or smaller layout area (Xiao et al., 2023; Feng et al., 2025; Tsai et al., 2025). Transistor sizing then adjusts transistor sizes within each topology to tune cell drive strength and PPA trade-offs according to design requirements (Xu et al., 2017). Therefore, optimization across size configurations, topologies, and logic functions creates a combinatorial design space of candidate pre-layout netlists. For each netlist, accurate PPA evaluation requires layout generation, design verification, parasitic extraction, and transistor-level simulation, as shown in Figure 1(b). At advanced technology nodes, restrictive design rules, limited routing resources, and parasitic effects further complicate physical implementation and PPA characterization, making accurate PPA evaluation across the netlist design space time-consuming (Xu et al., 2017; Vashishtha et al., 2017).

With advances in deep learning for electronic design automation, particularly the use of graph neural networks (GNNs) for circuit representation learning (Huang et al., 2021; Li et al., 2022; 2025a), prediction models have been developed to accelerate standard cell evaluation. Some existing methods combine netlist and layout information to predict standard cell performance for evaluating alternative layouts of a given netlist, but still require costly layout generation before prediction (An et al.,

[1] https://osf.io/seqjp/overview?view_only=dbdf13ff2a9f48bda7f9079311f37276

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

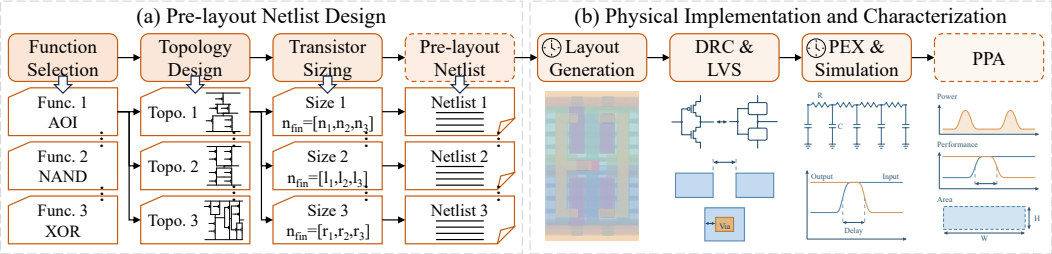


Figure 1: Standard cell library development, from logic functions, topology optimization, and transistor sizing to layout generation, verification, and PPA characterization.

2026; Zhang et al., 2026). Other methods use pre-layout netlists as inputs but focus on standard cell library characterization under varying electrical, process, and aging conditions (Ma et al., 2024; Liu et al., 2025; Jin et al., 2026). These studies do not systematically evaluate model generalization across the diverse size configurations, topologies, and logic functions present in the pre-layout netlist design space. Meanwhile, existing open circuit benchmarks, *e.g.*, CircuitNet and ArchPower, primarily focus on predicting PPA and physical design metrics at the circuit and architecture levels (Chai et al., 2023; Wang et al., 2026; Zhang et al., 2025), leaving a gap in data and evaluation protocols for standard cell-level prediction. Furthermore, although existing standard cell libraries contain commonly used logic functions and variants with different drive strengths, their coverage of the netlist design space is insufficient for this evaluation, and thus they provide only a starting point for data collection. Therefore, to advance machine learning-based prediction methods for exploring the netlist design space, we focus on generating netlists that differ from designs seen during training and aim to establish a benchmark for evaluating model generalization to unseen designs.

To this end, we propose CellBench, the first open dataset and benchmark for fast standard cell evaluation. We extend the ASAP7 7 nm standard cell library through a novel data generation pipeline to construct a diverse collection of 118,906 pre-layout netlists with post-layout PPA labels. For practical cell library optimization, a candidate netlist with favorable predicted PPA can improve circuit-level PPA only if it can be implemented as a layout. Thus, in addition to PPA prediction, we introduce constrained routability prediction to assess whether layout generation will succeed under specified tool settings and resource limits, using 296,274 netlists labeled with layout generation results. For each dataset, we construct three test sets to assess generalization beyond the training data, targeting unseen size configurations, topologies, and functions. By covering both prediction tasks across three generalization settings, CellBench supports the comprehensive evaluation of prediction models for standard cell library optimization at advanced technology nodes.

We benchmark existing standard cell prediction methods on CellBench. Evaluation results show that overall performance is best on unseen size configurations and worst on unseen functions. Further analysis of unseen functions shows that PPA prediction error is associated with topological similarity to training circuits, but not consistently with functional similarity. Our contributions are as follows:

- We introduce CellBench, to the best of our knowledge, the first open dataset and benchmark for fast standard cell evaluation, which covers PPA and constrained routability prediction and evaluates generalization to unseen size configurations, topologies, and functions, supporting standard cell library optimization at advanced technology nodes.
- We develop a data generation pipeline that combines logic function extension, topology generation, and size configuration assignment to construct diverse standard cell netlists.
- We evaluate existing standard cell prediction methods, observing declining overall performance across both tasks and three generalization settings. We further analyze functional and topology distances to understand differences in PPA prediction difficulty.

2 Preliminaries

Standard Cell Library. A standard cell library is a set of basic circuit components designed for a specific process technology. The library provides functional descriptions, transistor netlists, layouts, and timing and power models, which together support logic synthesis, placement and routing, simulation, and verification. A logic function can be implemented by multiple transistor topologies, and

Table 1: Comparison of input/output settings and generalization scenarios in studies related to standard cell PPA prediction. ✓ indicates that the study adopts the specified input/output setting or considers the generalization scenario; × indicates otherwise.

Task	Method	Input		Output	Generalization Scenarios		
		Pre-layout Netlist	Physical Layout	Post-layout PPA	Unseen Size	Unseen Topologies	Unseen Functions
Fast PPA Evaluation	EPiCell (An et al., 2026)	✓	✓	✓	×	×	×
	FusionCell (Zhang et al., 2026)	✓	✓	✓	×	×	×
Cell Library Characterization	ASPDAC'24 (Ma et al., 2024)	✓	×	✓	×	×	×
	DATE'25 (Liu et al., 2025)	✓	×	✓	×	×	×
	DATE'26 (Jin et al., 2026)	✓	×	✓	×	×	×

each topology admits multiple size configurations of its transistors; a topology together with a size configuration forms a pre-layout netlist, so function, topology, and size define a hierarchy of design decisions.

Library Development. Library development involves four main stages, as outlined in Figure 1. (1) *Circuit Design.* The required logic functions are specified within a cell architecture that constrains cell height and routing resources. Each function is implemented by a transistor topology, typically comprising complementary PMOS and NMOS networks in static CMOS logic. Transistor sizes are then selected to obtain suitable drive capability and timing behavior (Xu et al., 2017). (2) *Layout Generation.* The topology and sizes together define a pre-layout netlist, which is physically implemented through transistor placement and routing. (3) *Physical Verification.* Physical verification includes design rule checking (DRC) for compliance with manufacturing design rules and layout versus schematic (LVS) checking for consistency between the layout and netlist. (4) *Parasitic Extraction and Simulation.* Parasitic extraction (PEX) incorporates layout parasitics into the transistor netlist, which is then used in transistor-level simulations to characterize timing and power under specified process, voltage, and temperature (PVT) corner, slew, and load conditions.

3 Related Work

3.1 Standard Cell Performance Prediction

Recent work on accelerating standard cell library optimization through fast performance prediction has focused on exploring the cell layout design space. EPiCell (An et al., 2026) represents cell netlist and layout in a unified heterogeneous graph for cell performance prediction, while FusionCell (Zhang et al., 2026) uses cross-attention to fuse netlist and layout representations for multimodal prediction of cell timing and power. These methods accelerate the evaluation of layout candidates but still require layout generation, a major computational bottleneck in standard cell design. These methods therefore offer limited support for netlist optimization before physical implementation. Another line of work uses only pre-layout netlists as structural inputs and focuses on accelerating standard cell characterization under varying electrical, process, and aging conditions (Ma et al., 2024; Liu et al., 2025; Jin et al., 2026). However, predictive ability across characterization conditions does not establish generalization to unseen cell designs. CellBench evaluates this capability through explicit holdouts of size configurations, topologies, and functions. Table 1 summarizes the input/output settings and generalization scenarios considered in prior studies.

3.2 Benchmarks for Circuit Performance Prediction

Existing benchmarks for circuit performance prediction span multiple levels of design abstraction. At the architecture level, ArchPower (Zhang et al., 2025) studies processor power prediction from architectural configurations and execution statistics. At the logic level, OpenABC-D (Chowdhury et al., 2021) supports logic synthesis quality of results (QoR) prediction from AND-inverter graphs (AIGs) and synthesis recipes, while DeepCircuitX (Li et al., 2025b) enables pre-layout PPA prediction from register-transfer level (RTL) designs. Several benchmarks further incorporate layout information to support prediction across design stages. CircuitNet (Chai et al., 2023) provides netlists, placement, and power and timing features for predicting physical design metrics (*e.g.*, routing congestion and net delay). CircuitNet 2.0 (Jiang et al., 2024) expands the coverage to

Table 2: CellBench dataset statistics for PPA regression and constrained routability classification. Function and topology totals count distinct entries across splits.

	CellBench-1 PPA Regression			CellBench-2 Constrained Routability Classification				
Split	Functions	Topologies	Netlists	Functions	Topologies	Netlists	Success	Failure
Train	43	2,455	17,580	43	4,531	37,249	23,947	13,302
Validation	46	978	6,107	48	1,480	14,814	8,238	6,576
Test-1	43	2,222	11,878	43	4,471	25,031	16,106	8,925
Test-2	42	1,779	18,568	42	3,180	41,374	26,048	15,326
Test-3	50	8,865	64,773	50	17,366	177,806	85,718	92,088
Total	99	13,813	118,906	99	26,230	296,274	160,057	136,217

more diverse designs, while CircuitNet 3.0 (Wang et al., 2026) further connects RTL designs with their physical implementations to support multimodal timing and power prediction. However, existing benchmarks stop at the design-level and do not provide a benchmark for performance prediction of standard cells. CellBench fills this gap by pairing pre-layout netlists with post-layout PPA labels and further evaluating generalization to unseen designs.

4 CellBench

4.1 Dataset Overview

CellBench evaluates generalization in PPA and constrained routability prediction from pre-layout netlists of standard cells. Specifically, CellBench-1 covers 99 logic functions and 13,813 topologies, with 118,906 distinct netlists paired with five post-layout PPA targets. CellBench-2 covers the same 99 functions and 26,230 topologies, with 296,274 distinct netlists paired with binary constrained routability labels, as summarized in Table 2. All netlists in CellBench-1 also appear as successful samples in CellBench-2. Each sample pairs a netlist specifying transistor types, connectivity, and sizes with the corresponding labels. To evaluate generalization in both tasks, we partition each dataset into a shared training set, a validation set, and three test sets targeting unseen size configurations (Test-1), topologies (Test-2), and functions (Test-3).

Prediction Targets. To support cell candidate evaluation at an early design stage, CellBench provides five post-layout PPA targets, namely worst rise delay, worst fall delay, leakage power, internal energy, and area, and supports constrained routability prediction for physical feasibility assessment. Worst rise delay and worst fall delay are the maximum delays from an input transition to a rising or falling output. These maxima reflect timing bottlenecks and support fast evaluation of the timing impact of a cell. Leakage power measures static power consumption when inputs are held fixed and the cell has settled. Internal energy is a proxy for energy per transition, computed by subtracting the recorded load capacitor energy from the recorded switching energy and averaging over simulated scenarios. Area measures the placement footprint of a cell. Routability describes the feasibility of implementing the required interconnects within available routing resources while satisfying design rules. CellBench uses constrained routability to assess whether AutoCellGen successfully generates a layout under limited computational resources and fixed tool settings, with success labeled 1 and failure labeled 0. Measurement details are provided in Section 4.3 and Section C.2.

4.2 Netlist Generation

Following the standard cell netlist optimization flow illustrated in Figure 1(a), we expand the dataset through function extension, topology generation, and size configuration assignment.

Function Extension. The ASAP7 standard cell library provides 77 combinational logic functions, primarily covering relatively simple functions commonly used in circuit designs. To extend this coverage to more complex functions motivated by practical design needs, we perform standard cell library extension, combining the logic of multiple existing cells into a single specialized cell to improve PPA (Liang et al., 2022; Fu et al., 2025). Specifically, we apply TeMACLE (Fu et al., 2025) to generate candidate cells by merging groups of cells in gate-level netlists from the EPFL benchmark (Amarú et al., 2015). Given the emphasis on area minimization in standard cell layout synthesis (Chung et al., 2025), we select candidates based on area savings. We deduplicate can-

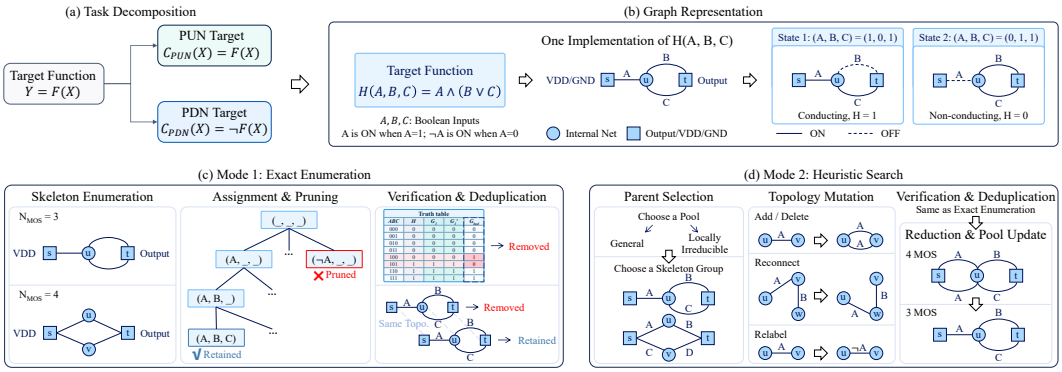


Figure 2: Overview of the topology generation algorithm: (a) task decomposition; (b) graph representation; (c) exact enumeration; (d) heuristic search.

didate functions and exclude those already present in ASAP7. The final benchmark includes 22 additional functions obtained through TeMACLE. The selected functions, together with functions from ASAP7, form the function set for subsequent topology generation.

Topology Generation. Evaluating generalization across topologies requires structurally distinct implementations of the same logic function. Existing topology generation methods formulate topology construction for a target function as a Boolean satisfiability (SAT) problem and enumerate solutions using SAT solvers (Xiao et al., 2023). As transistor count increases, the combinatorial growth in possible connections makes the search increasingly expensive, particularly for complex cells introduced through cell library extension. Although terminating the search early at a fixed time limit can still yield many implementation variants, these candidates may lack structural diversity. To address these limitations, we develop a graph search-based topology generation algorithm with two modes: exact enumeration to accelerate exhaustive search for small cells, and heuristic search to discover structurally diverse candidates within a limited time budget for larger cells.

As shown in Figure 2(a), our algorithm first decomposes topology generation for a function F into separate searches for pull-up networks (PUNs) and pull-down networks (PDNs). For a given combination of input values X, the PUN connects the output to power when $F(X)$ is one, while the PDN connects the output to ground when $F(X)$ is zero, corresponding to $C_{\mathrm{PUN}}(X) = F(X)$ and $C_{\mathrm{PDN}}(X) = \neg F(X)$, respectively. The resulting networks are paired to form complete CMOS topologies. Both searches follow two core steps.

(1) *Graph Representation.* As shown in Figure 2(b), each network is represented as an undirected multigraph: nodes represent electrical nets, and each edge represents a transistor. The output and power/ground nodes remain fixed during the search, while the remaining nodes represent internal nets between transistors. Each edge is labeled by a switch literal, *i.e.*, an input variable or its negation; the corresponding transistor conducts when the literal evaluates to one. The network state $H(X)$ is one if every transistor on at least one path from power/ground to the output conducts. A skeleton specifies only the connection structure. A labeled network further assigns a switch literal to each edge, determining when the corresponding transistor conducts. Different literal assignments can therefore produce distinct networks on the same skeleton.

(2) *Topology Search.* For small transistor networks, we use the exact mode (Figure 2(c)) to enumerate admissible skeletons within a specified transistor count range. For each skeleton, the algorithm searches for switch literal assignments that realize the target function, with pruning and symmetry reduction to avoid infeasible and redundant searches. Candidates are then checked against the target truth table, and equivalent networks are removed. For larger networks, we use the heuristic mode (Figure 2(d)). The search starts from reference networks constructed from binary decision diagrams (BDDs) and operates within a specified time budget and transistor count range. In each iteration, a parent network is selected for mutation by sampling a skeleton group and then one of its networks, balancing skeleton selection within each pool to encourage structural diversity. New candidates are generated by adding or deleting transistors, reconnecting edges, or relabeling edges. Candidates undergo functional and structural checks and deduplication before further exploration. Reductions that preserve the function produce additional candidates for subsequent search iterations.

After topology search, the verified PUNs and PDNs are paired to form complete CMOS cells. The assembled cells are checked again for logical correctness. Algorithm details are provided in Appendix B.1, and evaluation results in Section 5.4.

Size Configuration Assignment. Following topology generation, we assign transistor sizes to obtain pre-layout netlists. The ASAP7 PDK specifies a fixed gate length, leaving the integer fin count as the only adjustable size parameter (Clark et al., 2016). To generate diverse size configurations, we use multiple sizing strategies. For example, neighboring transistors sharing source/drain regions are required to have equal fin counts. For a subset of netlists, we therefore assign fin counts uniformly within each transistor type. The adopted 7.5-track cell architecture also limits each physical transistor segment to at most three fins (Xu et al., 2017). Accordingly, we assign fin counts that are multiples of three to selected transistors. Alternatively, we sample individual fin counts from weighted distributions, with a bias toward larger fin counts for output stage transistors in multi-stage cells.

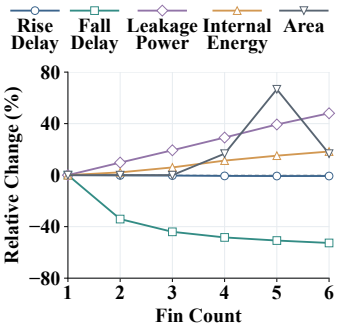


Figure 3: Ground truth PPA sensitivity to NMOS sizing in an OAI31 cell.

Simulating netlists for every size configuration is computationally expensive, so we sample a subset for each topology. To examine whether sparse samples can capture the main PPA trends, we conduct a series of sizing sweeps: irregular responses could leave important variation unrepresented in training. Figure 3 presents one OAI31 example in which the fin count of an NMOS transistor varies from 1 to 6, with the topology and all other transistor sizes fixed. Ground truth PPA values are expressed as percentage changes relative to the configuration with one fin. Worst rise delay, leakage power, and internal energy exhibit approximately linear trends; worst fall delay is approximately linear in the reciprocal of fin count, whereas area exhibits some nonlinearity. These trends motivate sampling to represent PPA variation with fewer simulations, rather than exhaustively enumerating size configurations.

4.3 Layout Generation and PPA Characterization

After assigning transistor sizes, we pass the resulting netlists to AutoCellGen (Baek & Kim, 2021) for transistor placement and routing. We use the same placement and routing settings in AutoCellGen and derive constrained routability labels from the routing results. We perform DRC and LVS checks on the generated layouts using Calibre, followed by parasitic extraction with xACT. The resulting post-layout netlists are then simulated using Spectre and ASAP7 regular threshold models under the same PVT conditions for all samples: the TT process corner, 0.7 V, and 25 °C. The same library buffer drives the switching input for all samples, with a 2.5 fF capacitor loading the measured output. For each output, we simulate all single-input transitions that trigger a rising or falling transition. We also measure leakage power as the mean cell supply power during the initial steady state of each simulation. Internal energy is computed as the recorded switching energy minus the recorded load capacitor energy. Leakage power and internal energy are averaged equally across scenarios to obtain the corresponding cell-level targets. Cell area is calculated as the layout width multiplied by the fixed row height. Further details appear in Appendices C.1 and C.2.

5 Evaluation

5.1 Experimental Setup

Dataset Splits. To evaluate generalization in PPA and constrained routability prediction, we partition CellBench-1 and CellBench-2 each into a training set, a validation set, and three test sets covering unseen size configurations, topologies, and functions. Test-1 contains unseen size configurations of topologies seen during training. Test-2 contains unseen topologies of functions seen during training. Test-3 contains unseen functions and all their implementations. Test-3 combines a subset of ASAP7 logic functions with all functions obtained through TeMACLE to evaluate generalization to complex logic functions. Note that functions are considered equivalent under input permutation, but output permutation and input or output negation are not allowed. See Appendix A.2 for splitting and model selection details.

Table 3: Regression and ranking performance on CellBench-1 across three generalization settings. Entries report MAPE (↓) / R^2 (↑) for regression and R_S (↑) / R_K (↑) for ranking. MAPE: %.

Test Set	Method	Worst Rise Delay	Worst Fall Delay	Leakage Power	Internal Energy	Area	Average
CellBench-1: PPA Regression (MAPE / R^2)							
Test-1	ASPDAC'24	18.48 / 0.67	17.05 / 0.64	3.74 / 0.98	6.28 / 0.97	8.97 / 0.85	10.91 / 0.82
	DATE'25	5.12 / 0.98	5.35 / 0.97	2.51 / 0.99	3.87 / 0.99	3.64 / 0.98	4.10 / 0.98
	DATE'26	4.71 / 0.98	4.97 / 0.98	2.17 / 0.99	3.43 / 0.99	3.29 / 0.98	3.71 / 0.99
Test-2	ASPDAC'24	19.13 / 0.53	17.52 / 0.59	4.52 / 0.98	7.04 / 0.96	8.87 / 0.85	11.42 / 0.78
	DATE'25	7.10 / 0.94	6.77 / 0.96	3.27 / 0.99	4.54 / 0.98	4.02 / 0.97	5.14 / 0.97
	DATE'26	6.32 / 0.95	6.34 / 0.96	2.94 / 0.99	4.31 / 0.99	3.62 / 0.98	4.71 / 0.97
Test-3	ASPDAC'24	19.49 / 0.49	20.22 / 0.45	8.79 / 0.94	9.59 / 0.93	8.79 / 0.85	13.38 / 0.73
	DATE'25	9.74 / 0.90	11.21 / 0.89	5.62 / 0.98	8.54 / 0.94	4.97 / 0.96	8.01 / 0.93
	DATE'26	9.72 / 0.91	10.34 / 0.90	5.77 / 0.97	7.35 / 0.96	5.08 / 0.96	7.65 / 0.94
CellBench-1: PPA Ranking (R_S / R_K)							
Test-1	ASPDAC'24	0.88 / 0.69	0.88 / 0.70	0.99 / 0.94	0.99 / 0.90	0.91 / 0.77	0.93 / 0.80
	DATE'25	0.99 / 0.92	0.99 / 0.91	1.00 / 0.96	1.00 / 0.94	0.99 / 0.93	0.99 / 0.93
	DATE'26	0.99 / 0.93	0.99 / 0.92	1.00 / 0.96	1.00 / 0.95	0.99 / 0.94	0.99 / 0.94
Test-2	ASPDAC'24	0.80 / 0.61	0.85 / 0.67	0.99 / 0.93	0.98 / 0.89	0.91 / 0.78	0.91 / 0.77
	DATE'25	0.97 / 0.87	0.98 / 0.89	1.00 / 0.95	0.99 / 0.93	0.99 / 0.93	0.99 / 0.91
	DATE'26	0.98 / 0.88	0.98 / 0.90	1.00 / 0.95	0.99 / 0.94	0.99 / 0.93	0.99 / 0.92
Test-3	ASPDAC'24	0.80 / 0.60	0.79 / 0.60	0.97 / 0.86	0.97 / 0.85	0.92 / 0.79	0.89 / 0.74
	DATE'25	0.95 / 0.81	0.94 / 0.79	0.99 / 0.91	0.97 / 0.87	0.98 / 0.90	0.97 / 0.86
	DATE'26	0.95 / 0.81	0.95 / 0.81	0.99 / 0.91	0.98 / 0.89	0.98 / 0.90	0.97 / 0.87

Evaluation Metrics. Fast standard cell evaluation serves different design needs: estimating PPA to quantify design improvements, and comparing candidate cells to guide selection. For regression, mean absolute percentage error (MAPE) measures the magnitude of relative prediction errors, while R^2 measures goodness of fit. For ranking, Spearman's rank correlation (R_S) measures agreement between predicted and reference ranks, whereas Kendall's rank correlation (R_K) measures agreement in pairwise ordering. Lower MAPE and higher R^2, R_S, and R_K indicate better performance. For each evaluation metric, Average denotes the arithmetic mean across the five PPA targets.

Baselines. We benchmark three representative methods that use pre-layout netlists for cell library characterization: ASPDAC'24 (Ma et al., 2024), GTN-Cell (Liu et al., 2025), and TRACE (Jin et al., 2026), as shown in Table 1. We adapt these methods to PPA regression and constrained routability classification using the netlist inputs provided by CellBench. The adapted methods are denoted as ASPDAC'24, DATE'25, and DATE'26. ASPDAC'24 represents each cell as a directed graph and applies a graph convolutional network for prediction. DATE'25 combines a bipartite graph representation with structural positional encodings and a graph transformer. DATE'26 represents each cell as a relational graph and uses a graph attention network with local and global adaptation for prediction. Adaptation details are provided in Appendix D.1. All baselines are trained with three random seeds using the same data splits. Hyperparameters and checkpoints are selected on the validation set, and test metrics are averaged across the three runs.

5.2 Main Benchmark Results

Table 3 reports PPA regression and ranking results. Across all three baselines, the overall trends in MAPE, R^2, R_S, and R_K indicate that both PPA value prediction and candidate ranking become progressively more challenging from unseen size configurations (Test-1) to unseen topologies (Test-2) and unseen functions (Test-3). Across all test sets, the baselines generally achieve poorer regression and ranking performance on worst rise and fall delays than on the other PPA targets, whereas they exhibit better prediction performance for leakage power. Compared with the other PPA targets, area generally shows smaller declines in regression and ranking performance from Test-1 to Test-3, which may be related to the direct dependence of area on layout geometry.

Table 4: Constrained routability prediction on CellBench-2 across three generalization settings. Success Recall, Failure Recall, Accuracy: %.

Test Set	Method	Success Recall	Failure Recall	Accuracy
Test-1	ASPDAC'24	80.40	74.73	78.38
	DATE'25	84.66	85.15	84.83
	DATE'26	84.65	86.25	85.22
Test-2	ASPDAC'24	75.64	78.99	76.88
	DATE'25	81.23	84.67	82.51
	DATE'26	81.06	85.68	82.77
Test-3	ASPDAC'24	64.37	84.51	74.80
	DATE'25	75.03	87.54	81.51
	DATE'26	72.20	88.74	80.77

Table 5: Function subgroup statistics for Test-3.

Subgroup	Functions	CellBench-1		CellBench-2	
		Topologies	Netlists	Topologies	Netlists
Same NPN Family	26	5,555	40,435	8,514	95,324
Different NPN Family (Near)	13	1,738	10,035	5,715	47,334
Different NPN Family (Far)	11	1,572	14,303	3,137	35,148

Table 6: PPA and constrained routability prediction on Test-3 and its function subgroups. CellBench-1 report mean MAPE / mean R^2 across the five PPA targets; CellBench-2 report Success Recall / Failure Recall / Accuracy. MAPE, Success Recall, Failure Recall, Accuracy: %.

Method	All Test-3	Same NPN Family	Different NPN Family (Near)	Different NPN Family (Far)
CellBench-1: PPA Regression (MAPE ↓ / R^2 ↑)				
ASPDAC'24	13.38 / 0.73	13.89 / 0.73	11.79 / 0.77	13.03 / 0.68
DATE'25	8.01 / 0.93	8.12 / 0.93	7.60 / 0.94	7.99 / 0.91
DATE'26	7.65 / 0.94	7.78 / 0.94	7.58 / 0.94	7.33 / 0.92
CellBench-2: Constrained Routability Classification (Success Recall / Failure Recall / Accuracy; All ↑)				
ASPDAC'24	64.37 / 84.51 / 74.80	60.99 / 83.97 / 71.92	60.87 / 88.45 / 79.49	75.35 / 77.60 / 76.30
DATE'25	75.03 / 87.54 / 81.51	76.69 / 84.94 / 80.61	67.12 / 92.14 / 84.01	76.94 / 85.57 / 80.57
DATE'26	72.20 / 88.74 / 80.77	72.42 / 86.71 / 79.22	65.98 / 92.86 / 84.13	76.36 / 86.06 / 80.45

For constrained routability prediction (Table 4), classification accuracy decreases from Test-1 to Test-3 across all models, following a similar trend to CellBench-1.

5.3 Generalization Analysis

The performance gap between Test-2 and Test-3 motivates further analysis. The baselines take netlist graphs as input without explicit logic function descriptions; thus, for these baselines, generalization to unseen functions is also a topology generalization problem. We therefore examine functional and topological differences from training and their associations with prediction difficulty.

Generalization Across Function Subgroups. We test whether greater logical similarity of test functions to training functions leads to more accurate predictions within Test-3. We first group test functions by NPN equivalence to training functions. For functions outside the training NPN families, we further quantify logical differences using functional distance d_{F}, defined as the smallest proportion of truth table entries that differ from a training function after NPN alignment. Accordingly, we divide Test-3 into three subgroups: functions from NPN families represented in training ($d_{\mathrm{F}} = 0$), functions from different NPN families at near distance ($0 < d_{\mathrm{F}} \leq 0.125$), and those at far distance ($d_{\mathrm{F}} > 0.125$), as summarized in Table 5. Table 6 shows no consistent decline in prediction performance as functional distance increases. In particular, the same NPN family group has higher mean MAPE and generally lower mean R^2 on CellBench-1, as well as lower accuracy on CellBench-2, than the near group. Thus, greater logical similarity to training functions does not necessarily imply easier prediction.

Function and Topology Distribution Shifts. Since functional distance alone does not explain the subgroup results, we examine both function and topology distributions relative to training. Although Test-2 contains only functions seen in training, the proportion of samples belonging to each function differs from training. For Test-2, the total variation distance from the training function distribution is 0.40 in CellBench-1 and 0.24 in CellBench-2. For Test-1, this distance is only 0.01 in both datasets. Thus, Test-2 differs from training in both topologies and function proportions, despite containing no unseen functions. We next quantify structural differences using topology distance, defined as the minimum cosine distance to a training topology based on local pattern counts from Weisfeiler–Lehman (WL) refinement. These counts preserve transistor types and terminal roles but exclude sizes and function labels. Although Test-3 has greater mean topology distance than Test-2 (Figure 4(a)), the far group within Test-3 has a lower mean topology distance than the other two NPN subgroups (Figure 4(b)). The results therefore do not show a consistent increase in topology distance with functional distance.

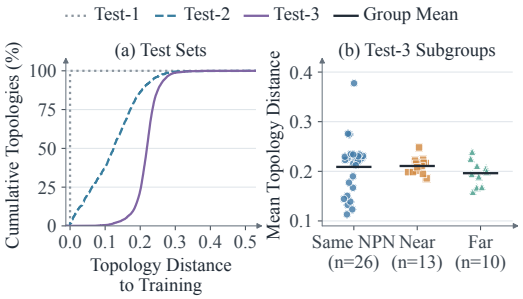


Figure 4: Topology distance in CellBench-1: (a) cumulative distributions of topology distance to training; (b) mean distance over distinct topologies within each Test-3 function.

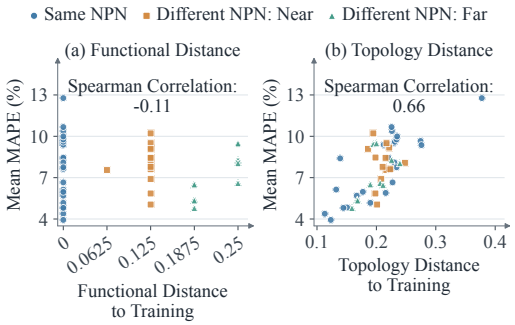


Figure 5: Functional and topology distance versus DATE'25 mean MAPE on CellBench-1 Test-3. Each point represents a function, with MAPE averaged over five PPA targets.

Topology Distance and Prediction Error. Since functional similarity alone does not explain prediction difficulty, we further examine the relationship between topology distance and prediction error. For each of the 49 Test-3 functions, we average topology distances over its distinct topologies. Prediction error is measured by MAPE averaged over the samples of that function and the five PPA metrics. All three models show positive correlations between topology distance and error ($R_S = 0.32$–0.66; Table 16), whereas correlations between functional distance and prediction error are weak or negative; Figure 5 presents DATE'25 as an example. The positive correlation between topology distance and prediction error persists even after controlling for transistor count, total fin count, input count, source composition, and functional distance (Appendix D.3, Table 16). Within individual functions, topology distance and prediction error also show a weak positive correlation on average, suggesting that differences between functions do not fully explain the overall relationship (Appendix D.3). Overall, these results suggest that topology similarity to training is more strongly associated with prediction error than functional similarity. Further details appear in Appendix D.3.

5.4 Topology Generation Results

We compare our proposed graph search-based method with MiniTNtk-Enum, an enumeration adaptation of SAT-based exact topology optimization method MiniTNtk (Xiao et al., 2023), in terms of search efficiency and structural coverage (Table 7). On the 34 small network instances with nonempty solution sets, our exact enumeration mode recovers the same complete solution sets as MiniTNtk-Enum and achieves a geometric mean end-to-end speedup of 7.85×.

For larger networks, we evaluate both methods on 15 logic functions with variable transistor counts and a 300 s time limit per run. We measure structural coverage by counting new skeletons of locally irreducible networks, which admit no further allowed reduction preserving functionality (Appendix B.1), excluding the common reference skeleton. Averaged over five runs per function and then across functions, our heuristic search mode produces 1,480.77 new skeletons, compared with 0.47 for MiniTNtk-Enum. Full results appear in Appendix B.4.

Table 7: Topology generation efficiency and structural coverage. Gain denotes geometric mean speedup or mean skeleton count increase.

Network Scale	Mode	Cases	Metric	MiniTNtk-Enum	Ours	Gain
Small	Exact Enumeration	34	Time (ms) ↓	1,529.24	40.54	7.85×
Large	Heuristic Search	15	New Skeletons in 300 s ↑	0.47	1,480.77	+1,480.31

6 Conclusion

We present CellBench, an open dataset and benchmark for fast standard cell evaluation. By combining function extension, topology generation, and transistor size assignment, CellBench expands beyond the implementations in the ASAP7 library and supports PPA and constrained routability prediction from pre-layout netlists. Evaluation across unseen size configurations, topologies, and functions exposes generalization gaps in existing methods. Further analyses show that structural similarity to training circuits helps explain PPA prediction error, whereas functional similarity alone does not consistently reflect prediction difficulty. Together, the data, evaluation protocol, and empirical findings support the development of prediction models for standard cell library optimization.

AI USE STATEMENT

Generative AI tools assisted with literature search, writing code for visualization and experiment automation, translation, and manuscript polishing. AI-assisted revisions were checked against the underlying research reports, experimental results, and source materials. Reported empirical findings are based on circuit simulation records and model evaluation results. The authors take responsibility for the final text, claims, and artifacts, including those developed with AI assistance.

REPRODUCIBILITY STATEMENT

The benchmark is available through the anonymous link in the abstract. Section 4 describes dataset construction, while Appendix A documents sample selection and the fixed training, validation, and test splits. Appendices B and C describe topology generation, physical design, and measurement procedures, and Appendix D describes baseline adaptations and additional analyses. Regenerating the physical measurements requires access to the stated EDA tools and technology files. However, the released netlists, labels, and data splits, together with the documented evaluation setup, allow the benchmark experiments to be reproduced without rerunning the physical design flow.

REFERENCES

Luca Amarú, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. The EPFL combinational benchmark suite. In *International Workshop on Logic and Synthesis (IWLS)*, 2015.

Wenbo An, Kairong Guo, Haoyi Zhang, and Yibo Lin. EPiCell: Electro-physical co-modeling for standard cell PPA prediction. In *Proceedings of the 63rd ACM/IEEE Design Automation Conference*, 2026. doi: 10.1145/3770743.3810437. URL https://kairongguo.com/files/DAC2026_CellChar.pdf.

Kyeonghyeon Baek and Taewhan Kim. Simultaneous transistor folding and placement in standard cell layout synthesis. In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–8. IEEE, 2021.

Zhuomin Chai, Yuxiang Zhao, Wei Liu, Yibo Lin, Runsheng Wang, and Ru Huang. CircuitNet: An open-source dataset for machine learning in VLSI CAD applications with improved domain-specific evaluation metric and learning strategies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023. doi: 10.1109/TCAD.2023.3287970.

Animesh Basak Chowdhury, Benjamin Tan, Ramesh Karri, and Siddharth Garg. OpenABC-D: A large-scale dataset for machine learning guided integrated circuit synthesis. *arXiv preprint arXiv:2110.11292*, 2021. URL https://arxiv.org/abs/2110.11292.

Sehyeon Chung, Hyunbae Seo, and Taewhan Kim. Synthesis of standard cells of minimum delay. In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–8. IEEE, 2025.

Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. ASAP7: A 7-nm FinFET predictive process design kit. *Microelectronics Journal*, 53:105–115, 2016. doi: 10.1016/j.mejo.2016.04.006.

Lang Feng, Rongjian Liang, and Hongxin Kong. Hybrid exact and heuristic efficient transistor network optimization for multi-output logic. In *Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7. IEEE, 2025.

Rongliang Fu, Chao Wang, Bei Yu, and Tsung-Yi Ho. TeMACLE: A technology mapping-aware area-efficient standard cell library extension framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 44(8):3034–3045, 2025.

Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Mingyuan Ma, Zhaoyang Shen, Juejian Wu, Yuanfan Xu, Hengrui Zhang, Kai Zhong, Xuefei Ning, Yuzhe Ma, Haoyu Yang, Bei Yu, Huazhong Yang, and Yu Wang. Machine learning for electronic design automation: A survey. *ACM Transactions on Design Automation of Electronic Systems*, 26(5):40:1–40:46, 2021. doi: 10.1145/3451179.

Xun Jiang, Zhuomin Chai, Yuxiang Zhao, Yibo Lin, Runsheng Wang, and Ru Huang. CircuitNet 2.0: An advanced dataset for promoting machine learning innovations in realistic chip design environment. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nMFSUjxMIl.

Muyan Jin, Chao Yang, Yunlin Liu, Zejian Cai, Pengpeng Ren, and Zhigang Ji. TRACE: A transferable framework for aging-aware cell delay estimation. In *Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7. IEEE, 2026.

Bohao Li, Qingsong Peng, Changhong Wang, Tianming Ni, Tinghuan Chen, Qi Sun, and Cheng Zhuo. H3Match: A hybrid heterogeneous hypergraph matching method for subcircuit identification. In *62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7, 2025a. doi: 10.1109/DAC63849.2025.11133089.

Min Li, Sadaf Khan, Zhengyuan Shi, Naixing Wang, Huang Yu, and Qiang Xu. DeepGate: Learning neural representations of logic gates. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 667–672, 2022. doi: 10.1145/3489517.3530497.

Zeju Li, Changran Xu, Zhengyuan Shi, Zedong Peng, Yi Liu, Yunhao Zhou, Lingfeng Zhou, Chengyu Ma, Jianyuan Zhong, Xi Wang, Jieru Zhao, Zhufei Chu, Xiaoyan Yang, and Qiang Xu. DeepCircuitX: A comprehensive repository-level dataset for RTL code understanding, generation, and PPA analysis. In *2025 IEEE LLM Aided Design Workshop (LAD)*, 2025b. URL https://arxiv.org/abs/2502.18297.

Tingyuan Liang, Jingsong Chen, Lei Li, and Wei Zhang. AutoCelllibX: Automated standard cell library extension based on pattern mining. *arXiv preprint arXiv:2207.12314*, 2022.

Lihao Liu, Yunhui Li, Beisi Lu, Li Shang, and Fan Yang. GTN-Cell: Efficient standard cell characterization using graph transformer network. In *Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7. IEEE, 2025.

Tianliang Ma, Zhihui Deng, Xuguang Sun, and Leilai Shao. Fast cell library characterization for design technology co-optimization based on graph neural networks. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 472–477. IEEE, 2024.

Jiun-Cheng Tsai, Wei-Min Hsu, Kuei-Lin Wu, Hsuan-Ming Huang, Jen-Hang Yang, Heng-Liang Huang, Yen-Ju Su, and Charles H-P Wen. MuSTNet: SAT-based exact multi-stage transistor network synthesis with placement awareness. In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–8. IEEE, 2025.

Vinay Vashishtha, Manoj Vangala, and Lawrence T Clark. ASAP7 predictive design kit development and cell design technology co-optimization. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 992–998. IEEE, 2017.

Mingjun Wang, Yihan Wen, Yuntao Lu, Fengrui Liu, Yuxiang Zhao, Boyu Han, Jianan Mu, Yibo Lin, Runsheng Wang, Bei Yu, and Huawei Li. CircuitNet 3.0: A multi-modal dataset with task-oriented augmentation for AI-driven circuit design. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lEDb4gQ4dB.

Weihua Xiao, Shanshan Han, Yue Yang, Shaoze Yang, Cheng Zheng, Jingsong Chen, Tingyuan Liang, Lei Li, and Weikang Qian. MiniTNtk: An exact synthesis-based method for minimizing transistor network. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 01–09. IEEE, 2023.

Xiaoqing Xu, Nishi Shah, Andrew Evans, Saurabh Sinha, Brian Cline, and Greg Yeric. Standard cell library design and optimization methodology for ASAP7 PDK. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017. doi: 10.1109/ICCAD.2017.8203890. URL https://arxiv.org/abs/1807.11396.

Haoyi Zhang, Kairong Guo, Bojie Zhang, Yibo Lin, and Runsheng Wang. Fusioncell: Cross-attentive fusion of layout geometry and netlist topology for standard-cell performance prediction. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=os54CnlmbR.

Qijun Zhang, Yao Lu, Mengming Li, Shang Liu, and Zhiyao Xie. ArchPower: Dataset for architecture-level power modeling of modern CPU design. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=shDB5inCJO.

APPENDIX

A DATASET AND EVALUATION PROTOCOL

A.1 SAMPLE IDENTITY AND DEDUPLICATION

The PPA dataset retains one record per canonical circuit from 125,672 measurements. Duplicate records are resolved by selecting the lexicographically smallest sample identifier, without using PPA, DRC status, or layout quality. Each retained circuit has a successful constrained routability record in the same split. Its labels describe the selected implementation, not the best attainable PPA of the netlist.

Circuit identity allows input permutation while preserving transistor types, sizes, connectivity, terminal roles, supply and output roles, and output order. Conservative isolation groups also link equivalent parallel transistor representations and related source records to prevent leakage across splits. Their physical implementations need not have identical labels.

A.2 SPLIT CONSTRUCTION AND VALIDATION

Hierarchical Allocation. All 22 TeMACLE functions are assigned to Test-3. Six functions are reserved for validation: NAND2 and OA221 share NPN families with training functions; A2O1A1I and O2A1O1I belong to different families in the near group; XNOR2 and XOR2 belong to different families in the far group. Selection uses NPN coverage, the validation budget, and stable hash ordering among candidates with at least 100 samples and ten original topologies, without consulting PPA labels or prediction errors.

Within training functions, approximately 40% of topology groups are assigned to Test-2, followed by additional groups for unseen topology validation. Within the remaining topologies, approximately 40% of size configuration groups are assigned to Test-1, followed by unseen size configuration validation. Each step preserves the required training support. The initial validation budget is 5% of samples; the 40% targets apply to groups, not global sample counts. CellBench-2 extends these assignments while keeping matched circuits in the same split, without class balancing or label-based seed selection. Filtering preserves split membership, and validation weights are recomputed for retained samples.

Isolation. The five splits are disjoint in sample identifiers, complete sized circuits, circuits normalized for parallel transistor representations, and conservative size configuration groups. Test-2 and unseen topology validation also exclude topology groups found in training or each other; Test-3 and unseen function validation similarly hold out functions. Test-1 and unseen size configuration validation may share training topologies, and fin counts need not lie outside the training range. Function identity allows input permutation but preserves output order and excludes input/output negation, so unseen functions can share an NPN family. Near and far groups have minimum NPN-aligned distances to training of ≤ 0.125 and > 0.125. Cases without training references of matching input/output counts are analyzed separately.

Validation and Model Selection. Validation covers unseen size configurations, topologies, and functions. For stratum s with weight α_s, F_s functions, and $n_{s,f}$ samples of function f, each sample receives weight $\alpha_s/(F_s n_{s,f})$. Functions within a stratum thus receive equal weight. Size and topology strata receive 0.2 and 0.4; each NPN function group receives 0.4/3. Weighted validation MAPE selects PPA hyperparameters and checkpoints, separately per target for ASPDAC'24 and equally across targets for joint models. Classification uses weighted validation binary cross entropy for model selection and weighted balanced accuracy for threshold selection. The threshold remains fixed across test sets. Model fitting and normalization use training data only.

All three training runs share the same splits, with model selection completed before test prediction. Earlier dataset versions informed development, and some historical test functions moved to validation. The current splits satisfy the isolation rules but were not held out throughout the entire research process.

A.3 Data Records and Usage

Model inputs contain transistor types, sizes, connectivity, and port roles. Function, topology, and isolation group annotations are used only for grouping and evaluation. Labels use seconds for delay, watts for leakage power, joules for internal energy, and square micrometers for area. Additional measurements retain their split assignments but are excluded from the main training set. Release documentation provides loading and evaluation instructions, fixed split memberships, and validation weights. Distributed inputs exclude layouts, extracted netlists, waveforms, commercial tools, and device models.

B Topology Generation

B.1 Graph Representation and Verification

The generator searches for a network with a prescribed state $H : \{0,1\}^n \to \{0,1\}$, where H is non-constant and input order is fixed. A candidate $G = (V, \mathcal{E}, s, t, \lambda)$ comprises electrical nodes V, transistor edges $\mathcal{E}$, distinct fixed terminals s, t, and switch literal assignments λ from $\Lambda = \{x_i, \neg x_i\}_{i=0}^{n-1}$. Parallel edges are allowed, but self-loops are not. For each input combination X, edges with literals equal to one must connect s to t if and only if $H(X) = 1$. The skeleton must be connected, each internal node must have degree greater than one, and every edge must lie on a simple terminal path.

Canonical colored incidence graphs identify networks while preserving terminal roles, switch literals, and edge multiplicity but ignoring internal names and edge order. Skeleton comparison omits switch literals. Neither comparison allows terminal exchange, input permutation, or complementation.

Candidate truth tables are computed by evaluating connectivity for all input combinations in parallel. Each generated network is independently rechecked by a separate graph traversal for each input combination.

For output function F, the PUN and PDN satisfy $C_{\mathrm{PUN}} = F$ and $C_{\mathrm{PDN}} = \neg F$. PMOS gate literals negate their switch literals, and required input inverters are shared. Assembled cells are checked for correct outputs and to exclude simultaneous connections to both supply rails. Transistor count limits apply to each PUN or PDN; multiple outputs require separate network construction. These ideal switch checks do not establish analog correctness or physical feasibility.

B.2 Exact Enumeration

The exact mode considers each specified transistor count m and node count $2 \le |V| \le m + 1$. A cached skeleton catalogue is built by adding edges, including parallel edges, or subdividing an edge with a new internal node, followed by structural checks and canonicalization. For each skeleton, a depth-first search assigns switch literals from candidate sets $D_e \subseteq \Lambda$ to its edges.

Pruning uses two connectivity bounds for each input combination. The *may* graph includes edges with at least one candidate literal equal to one; the *must* graph includes edges whose candidate literals all equal one. A branch is infeasible if $H(X) = 1$ but the may graph cannot connect the terminals, or if $H(X) = 0$ but the must graph already connects them. Testing literals against these bounds removes infeasible assignments. Search branches on the smallest candidate set with multiple literals and restores the sets when backtracking. Parallel edge ordering and skeleton symmetries reduce redundant exploration; labeled canonicalization removes remaining duplicates.

The transistor count lower bound L counts required literal polarities. If changing x_i from zero to one can raise H, the network requires x_i; if it can lower H, it requires $\neg x_i$. At $m = L$, each required literal occurs exactly once. This restriction does not apply when $m > L$.

Enumeration continues until the search is exhausted. Completeness requires a complete skeleton catalogue, sound pruning, and traversal without early termination. Experiments verify solution set agreement only within the tested finite spaces; agreement cannot rule out shared omissions. Interrupted searches cannot establish that no solution exists.

Algorithm 1 Heuristic generation of topology variants

Require: Target H, observation limit τ, random seed
Start timer; initialize identity set $\mathcal{S}$, output set $\mathcal{O}$, and parent pools
Construct and verify BDD references; determine L, U; accept eligible seeds
if no eligible seed is available **then**
    return $\mathcal{O}$ and No-Seed status
end if
while time remains **do**
    Select a parent pool with the stated 0.7 preference; fix its skeleton groups for this batch
    for up to 64 proposals within the remaining time **do**
        Sample a skeleton and parent; propose candidate G
        if G satisfies transistor count, truth-table, and structural constraints and has a new identity **then**
            Add the canonical identity of G to $\mathcal{S}$
            Reduce G greedily to G'
            Insert G into eligible parent groups, subject to capacity; record whether it is retained
            if G is irreducible or was retained in a group **then**
                Independently verify G; add it to $\mathcal{O}$
            end if
            if G' is smaller than G and time remains **then**
                Pass G' through the same acceptance procedure
            end if
        end if
    end for
end while
return $\mathcal{O}$ and Incomplete status

B.3 Heuristic Search

Initialization. The heuristic mode starts from BDD networks constructed using up to 256 variable orders. The eight smallest distinct candidates undergo greedy reduction; the smallest result defines the reference transistor count m_{ref}, which is not guaranteed to be globally minimal.

Parent Selection. The search supports up to eight inputs and transistor counts $L \leq m \leq U$, where $U = \min(16, \lfloor 1.5 m_{\mathrm{ref}} \rfloor)$. General and locally irreducible parent pools group networks by skeleton, retaining at most four labeled networks per group. Each batch of 64 proposals selects the irreducible pool with probability 0.7 when available, and the general pool otherwise. Within the selected pool, each proposal samples a skeleton group uniformly and then one of its networks uniformly. New skeleton groups become available in the next batch. Equal selection probability across groups avoids favoring structures with many literal assignments.

Graph Transformations. Candidates modify connections, switch literals, or transistor count. Transformations at fixed count reconnect or exchange edge endpoints, relabel edges or exchange their literals, combine deletion with subdivision, or split and merge internal connectivity. Growth duplicates an edge in parallel or subdivides it into two identically labeled edges; deletion and contraction produce smaller networks. Acceptance requires functional and structural verification and compliance with transistor count limits.

Reduction and Candidate Feedback. Greedy reduction repeatedly applies the first transistor deletion or legal edge contraction that preserves the truth table. Contraction removes self-loops but cannot merge the terminals. A locally irreducible network admits no further such reduction and requires every transistor to lie on a terminal path with simultaneously satisfiable switch literals. This does not imply minimum transistor count or favorable PPA. Algorithm 1 summarizes acceptance and reduction feedback.

Output Rules. Locally irreducible candidates are returned even when their skeleton groups are full; other candidates are returned only if retained in a parent pool. Reduced networks enter the search as additional candidates and do not automatically replace the original networks.

B.4 Evaluation Setup and Summary

MiniTNtk-Enum adapts the MiniTNtk CNF implementation (Xiao et al., 2023) for enumeration with the same terminal roles, switch literal domain, structural constraints, and output checks as our generator. Small-network runtimes include startup and input/output operations. For larger networks, coverage counts new locally irreducible skeletons generated within 300 s, excluding the shared reference skeleton. Reference construction counts toward this window; postprocessing and process completion may extend beyond it. The 15 larger functions were used for algorithm tuning, not held-out evaluation.

Our exact mode and MiniTNtk-Enum return identical complete solution sets for all 48 small test cases. Table 11 reports runtime ratios for the 34 cases with solutions and 14 with no solutions under the specified constraints.

No-Core removes both reduction feedback and the preference for irreducible parents. This reduces mean structural coverage, although individual functions can benefit (Table 8). The comparison does not isolate the contribution of either component.

Table 8: Structural coverage on 15 tuning functions. Wins, ties, and losses compare mean counts over five runs. Mean Difference is heuristic minus comparator, averaged over functions within each run and reported as mean ± sample standard deviation across runs.

Comparator	Wins	Ties	Losses	Mean Difference
MiniTNtk-Enum	12	3	0	$+1480.31 \pm 70.13$
No-Core	9	4	2	$+1348.79 \pm 67.63$

B.5 Detailed Results

The small test suite covers all 14 nonconstant functions of two inputs at transistor counts of one, two, and three, together with AND3, AND4, AO21, AO22, MUX2, and OR3 at counts of three, four, three, four, four, and three. Each configuration is run five times in randomized task order. Runs share eight physical CPU cores, so the reported timings reflect execution on a shared server.

Table 9 lists the larger functions and transistor count ranges. WTG5 denotes a weighted threshold gate. For the multiplexers, d_i is a data input and s is the index determined by two selector inputs.

Table 9: Functions in the generator evaluation suite. Counts refer to one switch network.

Function	Target H	m_{ref}	$[L, U]$
AND6	$\bigwedge_{i=0}^{5} x_i$	6	[6,9]
OR6	$\bigvee_{i=0}^{5} x_i$	6	[6,9]
AO222	$(x_0 \wedge x_1) \vee (x_2 \wedge x_3) \vee (x_4 \wedge x_5)$	6	[6,9]
OA222	$(x_0 \vee x_1) \wedge (x_2 \vee x_3) \wedge (x_4 \vee x_5)$	6	[6,9]
AOI33	$\neg[(x_0 \wedge x_1 \wedge x_2) \vee (x_3 \wedge x_4 \wedge x_5)]$	6	[6,9]
OAI33	$\neg[(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5)]$	6	[6,9]
Majority3	$\mathbf{1}[\sum_{i=0}^{2} x_i \geq 2]$	5	[3,7]
Majority5	$\mathbf{1}[\sum_{i=0}^{4} x_i \geq 3]$	13	[5,16]
Mux3	d_s For $s < 3$; 0 For $s = 3$	8	[7,12]
Mux4	d_s, $s \in \{0, 1, 2, 3\}$	11	[8,16]
Parity4	$\bigoplus_{i=0}^{3} x_i$	12	[8,16]
Parity5	$\bigoplus_{i=0}^{4} x_i$	16	[10,16]
Equal3	$\bigwedge_{i=0}^{2} (x_i \leftrightarrow x_{i+3})$	12	[12,16]
Mixed6	$[(x_0 \oplus x_1) \wedge (x_2 \vee x_3)] \vee (x_4 \wedge x_5)$	12	[8,16]
WTG5	$\mathbf{1}[2x_0 + x_1 + x_2 + 2x_3 + x_4 \geq 4]$	9	[5,13]

Table 10 reports new skeleton counts over five runs, omitting rows where all methods find none. The Average row weights all 15 functions equally within each run, including omitted rows, then summarizes across runs. Zero means no additional skeleton beyond the reference, not absence of an implementation. These comparisons are descriptive rather than tests of statistical significance.

All 480 runs on small networks completed. Our exact mode has a lower mean runtime for every test case, averaging 0.04 s across runs compared with 1.10 s for MiniTNtk-Enum. At this scale, process startup and interface overhead account for a substantial part of runtime.

Greater structural coverage comes with higher execution cost (Table 12). Heuristic search averages 386.08 s per run versus 295.40 s for MiniTNtk-Enum, with more postprocessing time and memory

Table 10: New locally irreducible skeletons within 300 s when transistor count can vary. Values are mean ± sample standard deviation over five runs; bold marks the highest mean, including ties.

Function	Transistor Count	Heuristic	MiniTNtk-Enum	No-Core
AO222	6–9	3.80 ± 0.45	0.00 ± 0.00	**4.00** ± 0.00
AOI33	6–9	0.60 ± 0.89	0.00 ± 0.00	**0.80** ± 0.45
Equal3	12–16	**2223.20** ± 261.18	0.00 ± 0.00	63.20 ± 4.09
Majority3	3–7	**6.80** ± 0.84	6.00 ± 0.00	6.00 ± 0.00
Majority5	5–16	**7755.20** ± 943.28	0.00 ± 0.00	1183.40 ± 68.80
Mixed6	8–16	**967.20** ± 455.65	0.00 ± 0.00	12.20 ± 0.45
Mux3	7–12	**1990.00** ± 347.03	0.00 ± 0.00	184.40 ± 12.03
Mux4	8–16	**3889.20** ± 206.09	0.00 ± 0.00	106.00 ± 28.56
OA222	6–9	**10.00** ± 0.00	1.00 ± 0.00	**10.00** ± 0.00
OAI33	6–9	**0.40** ± 0.89	0.00 ± 0.00	0.00 ± 0.00
Parity4	8–16	**438.20** ± 99.43	0.00 ± 0.00	107.80 ± 8.44
WTG5	5–13	**4927.00** ± 558.98	0.00 ± 0.00	302.00 ± 13.21
Average	—	**1480.77** ± 70.13	0.47 ± 0.00	131.99 ± 5.29

Table 11: Exact enumeration on small networks. Nonempty and Empty denote cases with and without valid solutions. Ratios are geometric means of paired MiniTNtk-Enum / Exact runtimes, weighting cases and runs equally.

Instances	Count	MiniTNtk-Enum / Exact
Nonempty	34	7.85
Empty	14	1.59

use. Coverage is thus compared within a common output observation window, not at equal total runtime.

Table 12: Execution cost on larger networks. Times average 75 runs per method. Max RSS is the maximum resident set size of an individual process, not the combined memory of concurrent processes.

Method	Process (s)	Search (s)	Postprocess (s)	Max RSS (MiB)
Heuristic	386.08	303.97	56.82	1793.24
MiniTNtk-Enum	295.40	295.30	0.04	487.60
No-Core	416.50	306.42	71.60	3059.35

Time limits are checked between operations, so running operations may finish after the observation window; later outputs do not count toward coverage. Process time includes orchestration and need not equal search plus postprocessing time. These experiments measure network discovery, not physical implementation quality or downstream PPA gains.

C Layout Generation and PPA Characterization

C.1 Layout Generation and Constrained Routability Labels

AutoCellGen uses ASAP7 (`asap7v3`) with the same folding, placement, and routing settings across recorded batches. Dynamic folding permits at most three fins per NMOS or PMOS segment. M1 allows horizontal and vertical routing, while M2 allows only horizontal routing. Usage of both layers is minimized without relaxing placement width.

Recorded time limits are 3,600 or 7,200 s per netlist; budgets and hardware are not established as uniform across attempts. Placement retains up to 50 candidates ranked by routing congestion cost, but only the first is routed. Collection prioritizes the `gds/` directory; if several outputs exist, the first in lexicographic path order is selected with its paired netlist.

Label 1 requires normal completion and a valid GDS with the expected top-level name. Label 0 requires normal completion, no GDS, and an explicit routing failure. Timeouts, execution errors, invalid outputs, and ambiguous records are excluded. Labels are independent of DRC status and describe tool outcomes under recorded settings; failure is not proof that no routable layout exists.

Equivalent netlist groups with conflicting labels from completed runs are excluded. Consistent duplicates use lexicographic ordering of batch and sample identifiers. PPA retains its own paired

implementation and follows Appendix A, rather than adopting the layout chosen during constrained routability deduplication.

C.2 Characterization Conditions and Scenario Enumeration

For the generated layouts, Calibre 2023.2_16.9 performs DRC and LVS, and xACT from the same installation extracts parasitics. Spectre 19.1.0.063 (64-bit) then simulates the extracted netlists under the conditions in Table 13.

For input vector $\mathbf{u}$, let $\mathbf{u}^{(i)}$ invert input i while leaving other inputs fixed. A simulation scenario is included for output j when $f_j(\mathbf{u}) \neq f_j(\mathbf{u}^{(i)})$. Both transition directions are simulated. Scenarios exclude simultaneous input switching and transitions that leave the measured output unchanged.

Table 13: Electrical and measurement settings used in characterization. The source edge time is distinct from the measured cell input slew.

Parameter	Setting
Device Models / Process Corner	ASAP7 RVT / TT
Supply Voltage / Temperature	0.7 V / 25 °C
Input Driver	`BUFx2_ASAP7_75t_R`
Measured Output Load	2.5 fF
Source Rise / Fall Time	40 ps / 40 ps
Source Transition Start / Simulation Duration	200 ps / 3 ns
Delay Threshold / Slew Thresholds	50% / 20–80% Of Supply
Initial Steady-State Window	0–100 ps
Footprint Width Unit / Row Height	54 nm / 270 nm

Only the measured output receives the explicit load capacitor; other outputs have no additional external loads. Separate cell and buffer supplies exclude buffer consumption from cell energy. Delay is measured between cell input and output crossings, excluding buffer delay.

C.3 PPA Target Definitions and Aggregation

Internal Energy. For simulation scenario α, let S_α denote the recorded switching energy (`e_switch_j`) and C_α the recorded load capacitor energy (`e_cap_j`). Internal energy is their mean difference over the scenario set $\mathcal{A}$:

$$e_{\text{int}} = \frac{1}{|\mathcal{A}|} \sum_{\alpha \in \mathcal{A}} (S_\alpha - C_\alpha). \tag{1}$$

This target is an energy proxy in joules per transition, not average power or a complete Liberty internal power characterization. No additional energy correction is applied during benchmark evaluation.

Delay, Leakage Power, and Area. Let $\mathcal{A}_\uparrow$ and $\mathcal{A}_\downarrow$ denote the scenarios with rising and falling outputs. For each scenario, $d_\alpha = t_{\text{out},\alpha} - t_{\text{in},\alpha}$ is the propagation delay measured at half the supply voltage, and P_α^{pre} is the mean cell supply power over the initial 0–100 ps window. Worst rise delay, worst fall delay, and leakage power are computed as

$$d_{\text{rise}} = \max_{\alpha \in \mathcal{A}_\uparrow} d_\alpha, \qquad d_{\text{fall}} = \max_{\alpha \in \mathcal{A}_\downarrow} d_\alpha, \tag{2}$$

$$p_{\text{leak}} = \frac{1}{|\mathcal{A}|} \sum_{\alpha \in \mathcal{A}} P_\alpha^{\text{pre}}. \tag{3}$$

Cell area is the layout width multiplied by the fixed row height: $N_{\text{CPP}} \times 0.054 \times 0.270\,\mu\text{m}^2$, where N_{CPP} is the recorded width in contacted poly pitches. Internal energy and leakage power assign equal weight to simulation scenarios. These means do not imply equal weighting of all input states or represent power consumption for a specific workload.

D Benchmark Experiments

D.1 Baseline Adaptations

GTN-Cell (DATE'25) replaces waveform prediction with five scalar PPA outputs. TRACE (DATE'26) retains its local and global branches but disables the original FiLM module and does not fine-tune on labeled test cells. For constrained routability, each baseline is trained separately with a binary classification output.

D.2 Ranking Within Functions

Standard cell library optimization selects among implementations of the same logic function. Correlations over all test cells also include comparisons between functions and therefore do not directly measure ranking quality for this selection task. Using the same baseline predictions, we compute R_S within each function for each PPA target and test set, comparing the available topology and size variants. Functions must have at least ten samples and nonconstant reference and predicted values. Correlations are averaged equally across eligible functions and then across the three training runs. The analysis includes 43, 42, and 48 functions in Test-1, Test-2, and Test-3, respectively.

Table 14 shows that the difference between global and within-function ranking depends on both the model and the PPA target. Delay correlations decrease most for ASPDAC'24, from 0.79–0.88 globally to 0.54–0.71 within functions, whereas DATE'25 and DATE'26 retain within-function correlations of 0.90–0.97. Leakage power and internal energy have high within-function correlations across all models and test sets, ranging from 0.94 to 0.99; area ranges from 0.83 to 0.97. Global correlations can therefore obscure weaker ranking within functions for some targets and models. The two summaries use different weighting and eligibility rules, and neither directly measures the PPA loss incurred when selecting a candidate.

Table 14: Ranking performance within logic functions on CellBench-1. Entries report global / mean within-function R_S (both ↑).

Test Set	Method	Worst Rise Delay	Worst Fall Delay	Leakage Power	Internal Energy	Area
Test-1	ASPDAC'24	0.88 / 0.59	0.88 / 0.68	0.99 / 0.98	0.99 / 0.95	0.91 / 0.83
	DATE'25	0.99 / 0.97	0.99 / 0.97	1.00 / 0.99	1.00 / 0.98	0.99 / 0.97
	DATE'26	0.99 / 0.97	0.99 / 0.97	1.00 / 0.99	1.00 / 0.99	0.99 / 0.97
Test-2	ASPDAC'24	0.80 / 0.54	0.85 / 0.71	0.99 / 0.98	0.98 / 0.95	0.91 / 0.84
	DATE'25	0.97 / 0.95	0.98 / 0.95	1.00 / 0.99	0.99 / 0.98	0.99 / 0.97
	DATE'26	0.98 / 0.96	0.98 / 0.96	1.00 / 0.99	0.99 / 0.98	0.99 / 0.97
Test-3	ASPDAC'24	0.80 / 0.64	0.79 / 0.69	0.97 / 0.96	0.97 / 0.94	0.92 / 0.84
	DATE'25	0.95 / 0.92	0.94 / 0.90	0.99 / 0.99	0.97 / 0.97	0.98 / 0.96
	DATE'26	0.95 / 0.93	0.95 / 0.93	0.99 / 0.99	0.98 / 0.98	0.98 / 0.97

D.3 Functional Distance, Topology Distance, and Prediction Error

Subgroup Evaluation and Weighting. Table 5 includes three PPA and 21 constrained routability samples without training references of matching input/output counts. These remain in All Test-3 but are excluded from subgroup metrics and distance analyses, leaving 49 PPA functions, including ten in the far group. Functional distance is the minimum fraction of differing truth-table entries after NPN alignment to a training function with matching input/output counts.

Subgroup comparisons depend on class and function composition. For example, DATE'25 has its highest accuracy but lowest balanced accuracy in the near group. Balanced accuracy, the mean of success and failure recall, is 80.81%, 79.63%, and 81.26% for the same NPN family, near, and far groups. Equal function weighting also changes DATE'25 PPA MAPEs to 7.86%, 7.98%, and 7.04%, shifting the lowest-error group from near to far. Functional distance alone therefore does not explain subgroup ordering.

Representations and Aggregation. Topology descriptors use separate transistor and electrical nodes, preserving PMOS/NMOS types, node roles, and drain, gate, source, and bulk port roles while excluding sizes and function identities. Local pattern counts from WL rounds zero through three form sparse vectors without compression into fixed buckets. This representation captures local connectivity but cannot distinguish all nonisomorphic graphs.

Topology distance is the minimum cosine distance to a training topology, averaged equally over distinct topologies within each function. Errors are computed per training run before averaging; function MAPE averages samples and the five targets equally. Correlation analysis weights the 49 functions equally. All baselines show positive topology correlations but weak or negative functional correlations (Figure 6). Association strength varies across models, so the DATE'25 example in the main text does not represent a uniform effect.

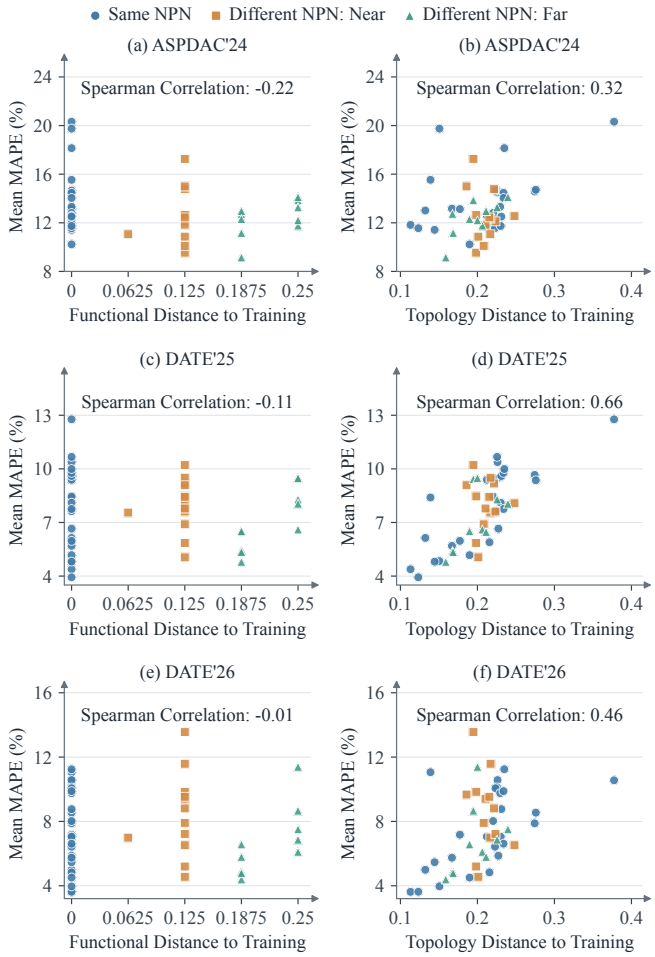


Figure 6: Functional and topology distance versus prediction error for all three baselines. Each point is a Test-3 function; all panels use the same 49 functions and the same aggregation as Figure 5.

Table 15: Distribution shifts relative to training. Function and NPN Family report total variation distance using netlist frequencies. Topology reports mean distance to the nearest training topology, averaged over distinct test topologies. All values are dimensionless.

Dataset	Test Set	Function	NPN Family	Topology
CellBench-1	Test-1	0.01	0.01	0.00
	Test-2	0.40	0.24	0.12
	Test-3	1.00	0.68	0.22
CellBench-2	Test-1	0.01	0.01	0.00
	Test-2	0.24	0.15	0.11
	Test-3	1.00	0.74	0.21

Distribution Statistics. Table 15 reports shifts relative to training. Total variation distance is half the sum of absolute category frequency differences, weighting netlists equally; topology means instead weight distinct topologies equally. Both tasks show similar function frequencies to training in Test-1 but shifts in Test-2, with greater mean topology distance in Test-3. Test-3 function identities are disjoint from training, but NPN families overlap. Figure 7 shows overlapping CellBench-2 topology distributions and no clear correspondence between functional and topology distance. Distances below 10^{-12} are treated as numerical zero, without implying graph or electrical equivalence.

Adjustment and Uncertainty. Partial rank correlations remove the linear effects of transistor count, total fin count, input count, and source composition from ranked distances and errors, then correlate the residuals. The final column of Table 16 also accounts for functional distance. Topology associations remain positive after adjustment.

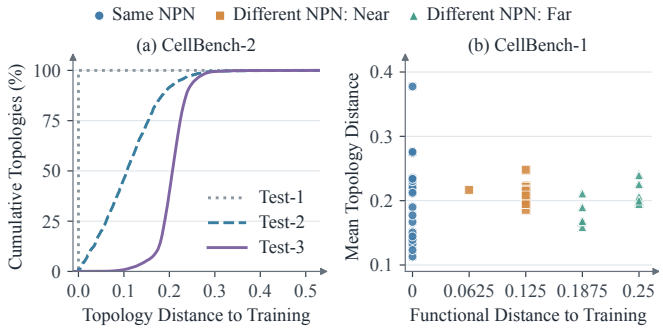


Figure 7: Supplementary distribution analysis. (a) Cumulative topology distances for CellBench-2, using the same axes and weighting as Figure 4(a). (b) Functional distance versus mean topology distance for the 49 CellBench-1 Test-3 functions; colors and shapes follow Figure 5.

Table 16: Associations between distance and mean MAPE per function. Functional Distance and Topology Distance report R_S; brackets give 95% NPN family bootstrap intervals for the correlation with topology distance. Adjusted columns report partial rank correlations, first using the controls described above and then also including functional distance.

Method	Functional Distance	Topology Distance (95% CI)	Adjusted	Adjusted + Functional Distance
ASPDAC'24	−0.22	0.32 [−0.04, 0.56]	0.34	0.31
DATE'25	−0.11	0.66 [0.34, 0.82]	0.73	0.72
DATE'26	−0.01	0.46 [0.07, 0.73]	0.71	0.73

Primary uncertainty estimates use 1,000 function bootstrap resamples. Table 16 uses 3,000 resamples of 23 NPN families to account for dependence within families. Intervals for unadjusted topology correlations remain positive for DATE'25 and DATE'26 but span zero for ASPDAC'24. Only DATE'25 has a positive interval for the difference between absolute topology and functional correlations, [0.04, 0.75]. Evidence for a stronger topology association thus varies across models. These exploratory percentile intervals are not corrected for multiple comparisons, and partial correlations do not establish causality.

Sensitivity to Reference Set and Source. Restricting training references to matching input/output counts reduces unadjusted topology correlations to 0.28, 0.24, and 0.06 for ASPDAC'24, DATE'25, and DATE'26. After adjustment for the same controls, they are 0.19, 0.37, and 0.46; function bootstrap intervals for the first two span zero. Association strength therefore depends partly on the reference set.

Source also matters: DATE'26 topology correlations are 0.56 on 27 standard library functions and −0.25 on 22 TeMACLE functions, while DATE'25 has corresponding correlations of 0.69 and 0.26. The aggregate relationship is therefore not uniform across sources. Replacing MAPE with mean absolute log error preserves positive unadjusted correlations of 0.64 and 0.44 for DATE'25 and DATE'26.

The far NPN group also has lower mean topology distance under one-round WL and the 2,048-bucket descriptor. However, the main three-round descriptor gives far-minus-same and far-minus-near function bootstrap intervals of [−0.0386, 0.0124] and [−0.0312, 0.0033]. Both span zero, so the observed ordering does not establish a general relationship.

Within-Function Analysis. For 48 functions with at least 20 samples and five topologies each, we compute partial rank correlations within each function, adjusting for transistor count, total fin count, and source. Mean correlations are 0.15, 0.12, and 0.10 for ASPDAC'24, DATE'25, and DATE'26, with positive correlations in 38, 39, and 35 functions. Their 95% intervals are [0.11, 0.19], [0.08, 0.16], and [0.06, 0.14], based on 2,000 function resamples rather than NPN family resampling. These weak positive associations suggest that differences between functions do not fully explain the aggregate relationship.

E Discussion and Limitations

CellBench explores the design space of standard cell pre-layout netlists but does not systematically evaluate alternative layouts of the same netlist. Although the source measurements can include multiple physical implementations, the main dataset retains one record per canonical circuit. At advanced technology nodes, restrictive design rules and limited routing resources complicate transistor placement and routing (Xu et al., 2017), making additional layout generation and verification costly. Alternative layouts can nevertheless reveal a broader range of attainable PPA trade-offs and implementations suited to different requirements. Extending CellBench with layout variants would support joint exploration of the netlist and layout design spaces.