GAPMA: Graph-Attention Cell Usage Prediction with Polarity-Minimized NPN Aggregation for Design-Specific Cell Composition

Chung-Kuan Cheng
UC San Diego
San Diego, USA
ckcheng@ucsd.edu

Junyeong Jang
UC San Diego
San Diego, USA
juj015@ucsd.edu

Andrew B. Kahng
UC San Diego
San Diego, USA
abk@ucsd.edu

Byeonggon Kang
UC San Diego
San Diego, USA
b8kang@ucsd.edu

Jakang Lee
UC San Diego
San Diego, USA
jal216@ucsd.edu

Abstract

Design-specific standard cell library augmentation can reduce post-route chip area, but selecting which cells to add is difficult before any cell layout exists. Prior methods rank candidates by *mining counts* that depend on the upstream synthesis flow, treat drive-strength variants as a single candidate, require completed cell layouts to evaluate area, and score candidates independently. We present GAPMA, an iterative framework that selects a small candidate-cell subset before any cell layout is generated. GAPMA aggregates mining counts across polarity variants of the same NPN class, estimates cell area from the transistor-level schematic via a common Euler path, and uses a graph attention network (GATv2) to predict cell usage shares jointly across existing and candidate cells. On five benchmark designs in an academic 3 nm PDK, GAPMA reduces post-route chip area by 14% to 30% (21% on average) relative to the baseline library.

CCS Concepts

• **Hardware** → **Physical design (EDA)**; • **Computing methodologies** → **Machine learning**.

Keywords

design-specific standard cell libraries, candidate cell selection, post-route replacement prediction, machine learning for EDA, candidate cell selection without candidate cell layout generation

ACM Reference Format:
Chung-Kuan Cheng, Junyeong Jang, Andrew B. Kahng, Byeonggon Kang, and Jakang Lee. 2026. GAPMA: Graph-Attention Cell Usage Prediction with Polarity-Minimized NPN Aggregation for Design-Specific Cell Composition. In *2026 ACM/IEEE International Symposium on Machine Learning for CAD (MLCAD '26), September 07–09, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3831599.3840351

1 Introduction

Commercial standard cell libraries are designed for broad applicability, but the logic functions most beneficial to implementation quality vary across designs. Identifying which cells are truly useful for a target design and developing a design-specific library accordingly is therefore at the core of design-specific standard cell library optimization. The resulting library contents directly affect synthesis, placement, and routing (SP&R) quality [1–3] over what a general-purpose library alone can achieve. When synthesis maps logic to gates, any function not covered by a single library cell must be implemented as a combination of smaller gates, which increases block-level cell-to-cell wiring complexity. A single added custom cell can absorb that wiring into its internal layout, potentially eliminating block-level routing without design-rule check (DRC) violations and reducing overall area. Such additions are called *candidate cells* [4–6]. The candidate cells that are most useful will vary across RTL designs and cannot be known a priori. Moreover, each new cell requires substantial effort for layout, signoff [7, 8] and PVT corner characterization [9] – and it is infeasible to build every candidate cell in advance.

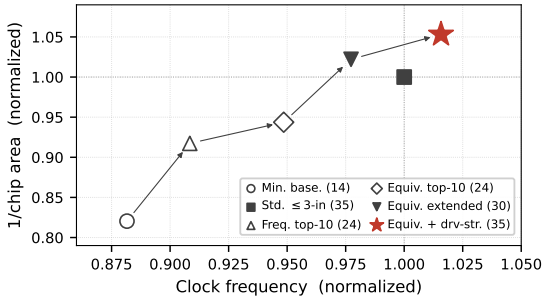

Figure 1: Best area–frequency operating points for the AES design, across six library configurations. Frequency is shown on the x-axis, and inverse area on the y-axis (upper-right is better). Both axes are normalized to the standard ≤3-input library (35 cells).

Pattern mining [10, 11] is a standard approach that scans the synthesized netlist for recurring gate patterns and ranks candidate cells by *mining count*, i.e., the number of times that each pattern appears. The implicit assumption is that a high mining count predicts high cell usage once the corresponding cell is added to the library. Unfortunately, this assumption fails [12–14]: a mining count records the synthesis decisions made under the original library, but synthesizing the same RTL with the augmented library produces a different gate mix. Prior methods [11–14] address aspects of this failing, but **four gaps remain**.

Gap 1. Mining counts depend on the library used in synthesis. The same RTL synthesized under two libraries will produce two different mining-count rankings, so a candidate ranked high under a given library can drop in rank once a new cell has been added. **Gap 2.** Drive-strength variants of the same function are ranked as a single candidate cell. Prior works [12–14] do not separate X1 and X2 candidates, so adding only one strength leaves the other-strength instances mapped to the baseline library as before; this reduces the realized benefit from the candidate. **Gap 3.** Cell-area reduction cannot be measured without generated cell layouts. The reduction equals the area of the gate pattern that synthesis would otherwise use, minus the candidate cell area. Prior methods read both numbers from existing layouts [7, 8], so a candidate without a generated layout never enters area-aware ranking. **Gap 4.** Cells in the library compete with each other for cell usage, but a per-cell scoring method does not account for this competition. When two cells implement overlapping logic, synthesis will assign a given substitutable location in the netlist to one cell or the other, so usage is shared. However, two cells that implement disjoint logic do not compete in this way. Hence, candidate scores must be evaluated jointly across the full augmented library.

Together, these gaps turn candidate-cell selection into a joint library-composition problem: each cell's benefit depends on which other cells are added with it. Our proposed GAPMA method addresses this joint composition problem by introducing polarity-minimized NPN (PM-NPN) mining-count aggregation for pattern mining, and graph attention network-based cell-usage-share prediction for candidate selection. Figure 1 previews the impacts of these elements for the AES design. In the figure, the x-axis is frequency and the

MLCAD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2878-5/2026/09
https://doi.org/10.1145/3831599.3840351

Table 1: Comparison of prior candidate-cell selection frameworks. S/F = structural/logic-function match. DS = drive-strength separated; IC = inter-cell competition; LB = learning-based selection; NL = no cell layout generation.

	Description			GAPMA contributions			
Framework	Match	Aggreg.	Objective	DS	IC	LB	NL
AutoCellLibX [11]	S	Pattern	Area	X	X	X	X
TeMACLE [12]	S	Pattern	ΔArea	X	X	X	X
SOFA-H [13]	F	P-class	ΔArea	X	△	X	X
CellE [14]	F	Pattern	QoR	X	△	X	X
GAPMA (Ours)	F	PM-NPN	Joint ΔArea	✓	✓	✓	✓

✓ yes, △ partial, X no.

y-axis is inverse area (better results are toward the upper-right). Each point represents the best operating point achieved by a given library configuration. The upward triangle and the diamond compare two mining strategies: the former adds 10 cells selected by frequency-based mining (24 cells total) while the latter adds 10 cells from our polarity-minimized NPN aggregation and outperforms the frequency-based selection. Then, the inverted black triangle further expands the library with six P-class-based cells (30 cells total), and the red star adds five X2 drive-strength variants to reach 35 cells total. Compared to the standard 35-cell ≤3-input library [15], the final library shares 24 cells with the baseline yet still attains a strictly better operating point.

Our main contributions are as follows.

- We propose PM-NPN mining-count aggregation, which sums mining counts over polarity variants of each NPN class.
- We formulate candidate-cell selection as joint cell-usage-share prediction over proposed library augmentations, performed by a graph attention network (GATv2).
- We evaluate GAPMA on five designs with the SO3 process design kit (PDK) [15] against the baseline library, a prior ≤3-input library, and an exhaustive 3-input library.

2 Background and Related Work

We review three background elements for our work: the candidate selection problem; the mining and aggregation units that define candidate functions; and prior selection frameworks that combine these elements in different ways.

2.1 Candidate-Cell Selection

Candidate-cell selection takes as input a current standard cell library L_i and an RTL design that has been synthesized, placed and routed with L_i. The goal is to augment the library with a small set of new cells, such that reimplementing the same RTL using the augmented library will improve quality of results (QoR). Prior frameworks ground this goal in *pattern mining*, which enumerates recurring gate patterns in the synthesized netlist and assigns to each such *candidate pattern* a *mining count* that records the number of times it appears in the netlist. Each candidate pattern, once realized as a single cell, has a corresponding *cell area*, the area of its physical layout. The selection target is to reduce the design's total cell area by replacing high-mining-count gate patterns with denser single (candidate) cells.

2.2 Pattern Mining and Functional Aggregation

Frequent pattern mining [10, 11] enumerates recurring bounded subgraphs in the gate-level netlist of an implemented design and records each pattern's mining count. A pattern can be counted structurally, where each distinct sub-netlist topology has its own key, or function, where a truth-table canonicalization maps structurally distinct but functionally equivalent patterns to the same key. The aggregation unit for functional mining is an equivalence class on Boolean functions, defined by which operations are allowed within the class. We distinguish two kinds of aggregation unit in this work. The *P-equivalence class* (or *P-class*) groups Boolean functions that differ only by input permutation; for example, $a \wedge b \wedge c$ and $b \wedge a \wedge c$ belong to the same P-class because one is a reordering of the inputs of the other. The 3-input Boolean functions partition into 80 P-classes, and the 4-input functions partition into 3,984 P-classes. The *NPN equivalence class* (or *NPN class*) is coarser, following the classical Boolean-matching formalism [16]. The name expands into the three operations it allows: **N**egate inputs, **P**ermute inputs, **N**egate output. Two functions are NPN-equivalent if one can be obtained from the other by some combination of these operations, and the members of one NPN equivalence class are its *polarity variants*. For example, $a \wedge b \wedge c$ and $!a \wedge !b \wedge !c$ are NPN-equivalent because negating all three inputs maps one to the other; the two are not P-equivalent because P-equivalence does not allow input negation. In summary, the aggregation unit determines whether such polarity variants are counted together (under NPN) or separately (under P-class).

2.3 Prior Candidate-Cell Selection Frameworks

We now summarize mining and selection properties of prior frameworks; see also Table 1. AutoCellLibX [11] propose the frequent-subgraph-mining approach with a pattern-combination algorithm that iteratively selects gate-level patterns to maximize total area reduction. TeMACLE [12] adds technology-mapping awareness to the mining stage with K-feasible cone extraction and SAT-based exact subcircuit matching. SOFA-H [13] mines fanout-induced multi-output hypercells with P-Representative canonical encoding, then jointly selects and remaps cells via a one-shot weighted MaxSAT formulation. CellE [14] replaces single-netlist mining with equality saturation over an e-graph of functionally equivalent implementations, to discover composite cells inaccessible to structural traversal. Bhattacharya et al. [17] automatically synthesize multiple design-specific candidate-cell transistor topologies and rank them by a pre-layout cost function before any physical layout is generated. Orthrus [18] addresses cell optimization within a broader system-technology co-optimization loop and is therefore complementary to candidate-cell selection. Lyn et al. [19] intervene after placement, replacing small groups of two to three adjacent cells with a single functionally equivalent standard cell to reduce timing-violation delay. Yoon et al. [20] similarly intervene after placement, but instead select adjacent cell pairs by maximum-weighted matching and eliminate the margin between them entirely.

3 Proposed Approach

Our framework iteratively selects design-specific candidate cells. At each iteration, it runs three stages after obtaining a post-route design with the current library: (1) PM-NPN mining-count aggregation filters the candidate pool; (2) pre-layout cell area estimation quantifies each candidate's area without cell layout generation; and (3) joint prediction by a graph attention network (GATv2) [24] selects the k_2-subset minimizing the total predicted cell area while capturing inter-cell usage competition. Table 2 lists notation used in the following discussion.

3.1 Overall Framework

Figure 2 presents the three-stage flow. Iteration i begins with the post-route design D_i from the previous SP&R run on library L_i. (1) Logic-cluster mining on D_i yields the top-k_1 functions by mining count, forming the first-stage filter. The mining records counts per P-class variant, and the PM-NPN aggregation combines counts across all P-class variants sharing the same PM-NPN representative so that each function's true cell usage share is exposed. Each selected PM-NPN function is paired with its available drive-strength variants, forming the candidate-cell pool C. (2) The pre-layout cell-area proxy $\hat{a}_i$, equal to the candidate's poly count, is computed from the transistor-level schematic without cell layout generation. (3) The third stage applies a GATv2 model trained on SP&R runs. For each size-k_2 subset $S \subseteq C$, the framework constructs a hypothetical next library $B_i(S) = L_i \cup S$ and

Table 2: Notation used in our discussion. Subscripts i index cells in $L_0 \cup C$ unless D_i or L_i are used as iteration indices.

Symbol	Meaning
Inputs and pool sizes	
L_0	Baseline standard cell library
D_0	Initial post-route design from L_0
D_i	Post-route design at iteration i
L_i	Library at iteration i
k_1	First-stage filter size; top-k_1 functions by mining count
k_2	Second-stage filter size; k_2 candidate cells selected by the GATv2
m	Total number of framework iterations
Mined functions and candidate cells	
f	Mined logic function; PM-NPN representative as defined in Sec. 3.2
$s \in \{\text{X1}, \text{X2}\}$	Drive-strength variant associated with a candidate cell
$c_{f,s}$	PM-NPN candidate cell with function f and drive strength s
C	Candidate-cell pool formed by pairing the top-k_1 mined functions with available drive-strength variants
Prediction and selection	
$\hat{a}_i$	Pre-layout cell-area proxy of cell i, equal to its poly count from a common Euler path on the schematic
$\bar{a}_i$	Average poly count of logic-cluster instances implementing i's function in the design D_{i-1} from the previous iteration
$B(S)$	Hypothetical next library $= L_{i-1} \cup S$ formed from a candidate subset $S \subseteq C$, used to score S during selection
$\hat{u}_i$	Predicted cell usage share of cell i in B's post-route netlist (per-graph softmax over the cells legal in B, with $\sum_{i \in B} \hat{u}_i = 1$)
$C^{k_2} \subseteq C$	Selected candidate subset (size k_2); $L_i = L_{i-1} \cup C^{k_2}$

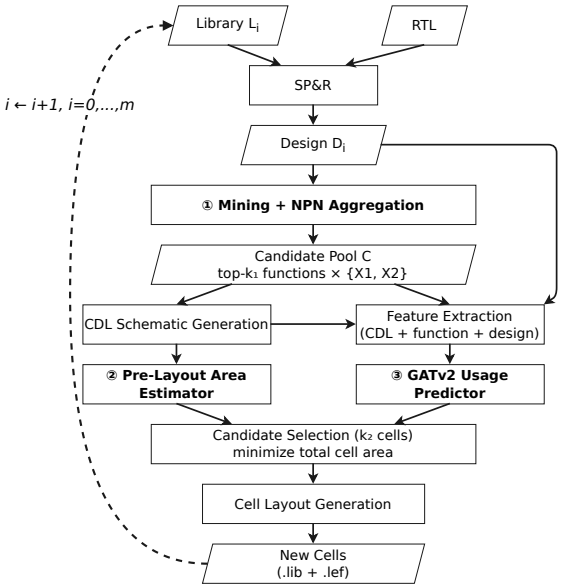

Figure 2: Proposed candidate-cell selection framework, iterated m times (dashed oval). Stages ①–③ are detailed in Sections 3.2–3.4.

predicts the cell usage share $\hat{u}_i(S)$ of each cell that would be available under $B_i(S)$. The framework selects the subset that maximizes the joint predicted area reduction; this objective is formalized below in Section 3.4.

X1+X2 inclusion rule. Within C, X1 and X2 variants of the same function compete as separate candidates, but the framework binds them at selection time: when the GATv2 picks $c_{f,\text{X1}}$ into C^{k_2}, $c_{f,\text{X2}}$ is also included whenever $\hat{u}_{c_{f,\text{X2}}} \geq 0.1 \cdot \hat{u}_{c_{f,\text{X1}}}$. With k_2 selected functions, the realized library augmentation contains between k_2 and $2k_2$ cells. Higher drive-strength variants are used mainly in timing optimization that serves critical-path instances: a function's X2 usage share is unlikely to exceed its X1 usage share, and binding X2 to X1 reflects this relationship.

The actual area change is measured by running SP&R on the selected library. Because k_1 and k_2 are small in the selection stage, evaluating these subset graphs is inexpensive relative to SP&R and cell layout generation. Our framework generates cells only for the k_2 selected candidates. An SP&R run with $L_i = L_{i-1} \cup C^{k_2}$ produces D_i, and the loop iterates by re-running stages (1)–(3) on D_i.

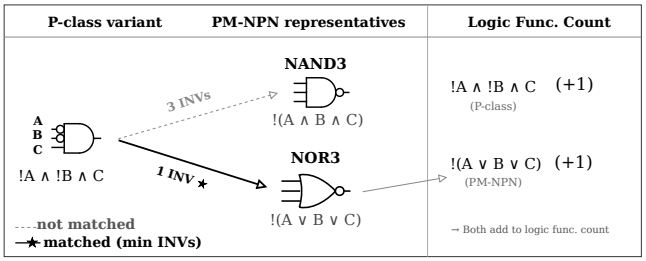

Figure 3: PM-NPN association. Each P-class variant maps to the PM-NPN representative requiring the fewest added inverters. For $f = !A \wedge !B \wedge C$, NOR3 requires one inverter (★) versus three for NAND3, so f maps to NOR3.

3.2 Mining and PM-NPN Aggregation

The mining problem we address is a form of frequent subgraph mining [21]. Our implementation (see [26] for source code) mines a specific class of patterns called *logic clusters*, which are bounded single-root subgraphs enumerated by depth-first search over each root's fanin cone, where each *root* is an output cell and its *fanin cone* is the set of cells whose outputs eventually drive that root, under the constraints η (logic-cluster cell count, input count, and logic depth).[1] A single-root logic cluster can have multiple boundary outputs through internal cells, and the miner additionally groups compatible single-root logic clusters to merge overlapping logic clusters and multi-output combinations that DFS alone does not capture. The mined set T_i retains overlapping logic clusters as competing coverage options for downstream selection.

A *polarity-minimized NPN representative (PM-NPN)* of an NPN equivalence class is the polarity variant requiring the fewest added inverter cells to realize the function as a single CMOS complex gate. Each mined polarity variant f is mapped to the PM-NPN of its equivalence class. Concretely, FindPMNPN(f) enumerates all input-inversion masks and output-flip choices, applies the P-canonical form, defined as the lex-smallest truth table under all input permutations, to each variant, and returns the minimum-cost variant whose complement is positive unate. Positive unate means flipping any input from 0 to 1 cannot flip the output from 1 to 0, which guarantees realizability as one naturally-inverting CMOS complex gate, without internal inverters.

Aggregating mining counts at PM-NPN granularity is more stable than at polarity-variant granularity because synthesis uses only the polarity variant available in L_0. The mining count at polarity-variant granularity is therefore split among the variants of the same class and is zero for any variant absent from L_0, even when that function is synthesized via NPN-equivalent cells already in the library. PM-NPN aggregation also preserves drive-strength flexibility because when synthesis realizes a non-minimized polarity variant, it inserts boundary inverters drawn from the drive-strength range already in L_0, so each inverter can be independently sized to match local fanout and timing slack. Selecting the PM-NPN at ranking time, rather than fixing a specific non-minimized polarity variant, preserves this freedom. The drive strength of both the boundary inverters and the candidate cell itself is resolved at technology-mapping time, not at library-construction time.

Coarse filter to candidate pool C. The per-PM-NPN mining-count table $\{(f, \text{count}[f])\}_{f \in F}$ yields the top-k_1 entries as the candidate pool $C = \{c_{f,s} : f \in F_{k_1},\ s \in \{\text{X1}, \text{X2}\}\}$. Larger k_1 broadens the pool at the cost of more pre-layout area computation and GATv2

[1] All experiments fix $\eta = (2\text{–}8,\ 2\text{–}4,\ 3)$, bounding each logic cluster to 2–8 cells, 2–4 inputs, and depth 3.

Table 3: Mining count (in D_0) vs. post-route instance count (in D_1 and D_2) across three library configurations.

Logic function	#mined in D_0	Rank	#in D_1	#in D_2	Δ
$!(A_1 \vee !A_2 \vee !A_3)$	3848	1	452	200	−252 (−56%)
$!(!A_1 \wedge !A_2 \wedge A_3)$	2821	2	2782	408	−2374 (−85%)
		⋮			
NAND3	73	16	—	1835	+1835
NOR3	41	19	—	847	+847

inference over a larger graph. Smaller k_1 may exclude cells that rank highly under inter-cell usage competition.

3.3 Pre-Layout Cell Area Estimator

For each candidate $c_{f,s}$, our CDL synthesizer turns the truth table of f into a transistor-level series-parallel netlist; the same synthesizer accepts any truth table, not only PM-NPN representatives. We then use the poly count of this netlist as the pre-layout cell-area proxy. The poly count is the number of poly columns obtained by finding a common Euler path between the NMOS and PMOS networks (cf. the Euler-path formulation of single-row CMOS layout [22, 23]), which is computed as the longest common subsequence (LCS) of their finger-expanded gate sequences:

$$\hat{a}_{c_{f,s}} = F_n + F_p - \max_{\sigma_n, \sigma_p} \mathrm{LCS}\left(\hat{g}_{\sigma_n}, \hat{g}_{\sigma_p}\right). \tag{1}$$

Here, F_n and F_p are NMOS and PMOS finger counts, and $\hat{g}_\sigma$ is the gate-name sequence of ordering σ. The poly count is computed from the schematic alone, without cell layout.

3.4 Cell Usage Prediction via Graph Attention

As noted above, prior works such as [11–13] each estimate candidate-cell usage *independently*. This breaks down whenever multiple candidates co-exist because a newly added cell absorbs cell usage share from any cell in its NPN class that covers the same logic-cluster sites. Table 3 provides a motivating illustration with AES implementations across three library configurations. D_i is the post-route design implemented with library L_i. Mining counts are taken from D_0 under the baseline library L_0. $L_1 = L_0 \cup \{\text{top-10 mined candidates}\}$ and $L_2 = L_1 \cup \{\text{NAND3, NOR3, AOI21}\}$. "Rank" is the position in the mining output sorted by mining count. A dash marks functions absent from L_1. "Δ" reports the cell usage in D_2 minus the cell usage in D_1, with the relative change in parentheses. Selecting a candidate *set* that maximizes joint area reduction is therefore a *joint subset-selection* problem [25]: the framework seeks an item subset to maximize a downstream objective that is evaluated jointly over all selected items.

In GAPMA's stage 3, we use a GATv2 model [24] to predict cell usage shares for a proposed library augmentation.[2] The library-transition graph takes a pair of cell sets (A, B) as input: A contains the cells used in the current post-route implementation under L_{i-1}, and B contains the cells available in a hypothetical next library $B_i(S) = L_{i-1} \cup S$ for one candidate subset S. The directed graph

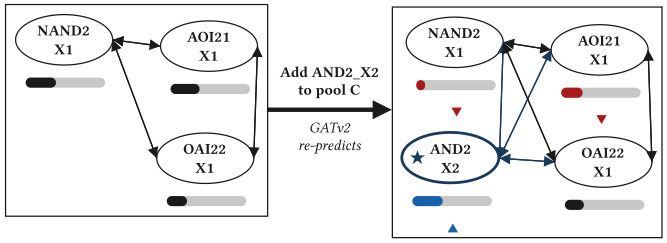

Figure 4: GATv2 dynamic cell-usage-share redistribution. Adding a candidate (★) triggers joint re-evaluation of every cell's cell-usage-share estimate $\hat{u}_i$ (bar length). ▼ = cell-usage-share decrease, ▲ = cell-usage-share increase.

[2]GATv2 is the fine refinement stage of a coarse-to-fine pipeline, where mining provides the coarse ranking by raw pattern frequency.

Table 4: Node (48-dim) and edge (7-dim) features used by the GATv2. "stats" denotes (avg, min, max). See our released code for full definitions [26].

Source	Group	Dim	Features
Node	CDL	9	#inputs, #NMOS, #PMOS, NMOS/PMOS fin counts, fin ratio, NMOS/PMOS stack depth, min poly count
	Truth-table	9	On-set weight, algebraic-normal-form (ANF) degree, affine/symmetric/monotone/ anti-monotone flags, in-inv count, out-inv count, transistor count
	Design	23	in-design flag, instance fraction, fanout stats, timing-violation ratio, path coverage, worst-slack rank, output-cap stats, input-slew stats, output-slew stats, dynamic-power stats, mining rank, poly-count-stack delay proxy, A-instance rank
	Library role	3	baseline-only, shared, candidate-only
	NPN rep.	4	PM-NPN flag, in-inv erased, out-inv erased, NPN-class size
Edge	Truth-table	4	same truth table, same NPN class, in-inv diff, out-inv diff
	Mining	3	co-occurrence ratio, relative count, overlap ratio

$G_S = (V_S, E_S)$ contains every cell that is either used in A or available in $B_i(S)$. Complete directed edges connect all ordered cell pairs, and edge features describe functional and mined substitutability (Table 4). The GATv2 therefore conditions each prediction on both the current design state and the cells that would be legal in the proposed next library. Within this framework, mining produces a top-k_1 candidate pool C by raw mining count. For a fixed selection budget k_2, the framework evaluates candidate subsets $S \subseteq C$ with $|S| = k_2$ by constructing the corresponding library-transition graph $(A, B_i(S))$ and predicting the cell usage shares $\{\hat{u}_i(S)\}$. The GATv2 uses the co-selected subset to make each prediction, so inter-cell usage competition is captured by the graph state (Figure 4).

Input. Nodes are cells; the graph is complete and directed, with every ordered cell pair (p, q), $p \neq q$, carrying an edge. The GATv2 learns substitutability from edge features rather than from a hand-tuned cutoff. Each node carries a 48-dimensional feature vector in five groups (CDL/structural, logic/truth-table, design context, library-role, NPN representative). Each directed edge carries a 7-dimensional feature vector in two groups (truth-table, mining). Table 4 lists the full set of features; Appendix A expands the per-group semantics.

Output. The GATv2 has two heads. The *distribution head* emits one logit per node, masks the logits of cells absent from $B_i(S)$ to $-\infty$, and applies a per-graph softmax to produce cell usage shares $\hat{u}_i(S)$ with $\sum_{i \in B_i(S)} \hat{u}_i(S) = 1$. The *scale head* gathers graph-level embeddings and applies a linear layer followed by softplus to predict the total-cell-count ratio $\hat{s} = \sum_{j \in B} c_B[j] \,/\, \sum_{j \in A} c_A[j]$, so the distribution head captures how cell usage redistributes across the library while the scale head captures whether the total cell count expands or contracts. Within this framework, mining produces a top-k_1 candidate pool C by raw mining count. For a fixed selection budget k_2, the framework evaluates candidate subsets $S \subseteq C$ with $|S| = k_2$ by constructing the corresponding library-transition graph (A, B) and predicting the cell usage shares $\{\hat{u}_i(S)\}$. The selected subset C^{k_2} maximizes the joint predicted area reduction:

$$C^{k_2} = \arg \max_{\substack{S \subseteq C \\ |S| = k_2}} \sum_{i \in S} \hat{u}_i(S) \cdot (\bar{a}_i - \hat{a}_i), \tag{2}$$

where $\bar{a}_i$ is the average poly count of logic-cluster instances implementing i's function in the current design (computed from mining), and $\hat{a}_i$ is the pre-layout poly count of cell i. The product $\hat{u}_i \cdot (\bar{a}_i - \hat{a}_i)$ is the expected area reduction from the predicted usage share of cell i. The maximized score serves as a ranking score over candidate libraries rather than an absolute area-change estimate; the actual area change is measured by SP&R after the candidates are realized.

3.5 GATv2 Training

Each training sample is an ordered pair of cell sets (A, B) derived from two SP&R runs of the same benchmark under different library subsets. The input graph uses only the current implementation A and the pre-route library membership of B, while the target u_i for cell i is its post-route instance count in B, normalized by the total logic-cell count of B. The network is a GATv2 that consumes the node and edge features of Table 4. The training labels are the per-cell usage share $u_i = c_B[i]/\sum_{j\in B} c_B[j]$ for the distribution head and the total-cell-count ratio $\sum c_B/\sum c_A$ for the scale head, both derived from post-route instance counts. The loss sums the cross-entropy on the distribution head, the mean squared error (MSE) on the scale head, and a pairwise margin loss on cell rank. Details of architecture, loss weights, and optimizer settings are given in Appendix B.

3.6 Training Data Generation

GATv2 is trained from sampled library-configuration runs. Each run provides a post-route implementation and a known allowed-cell set. Ordered pairs of runs (A, B) are then formed within the same design. A supplies the current implementation features, while B supplies the hypothetical next-library membership and the cell-usage-share target.

Training candidate set and sampled configurations. For each training design, an initial SP&R run under a fixed training baseline (INV, BUF, NAND2_X1, NOR2_X1 and DFF cells) produces a post-route netlist. Logic-cluster mining on this netlist yields the mining count of each canonical function. The per-design training candidate set is then formed by weighted random sampling of cells, using each function's mining count as its sampling weight, so more frequent functions are proportionally more likely to be included, and the distribution of sampled candidates reflects the design's observed logic mix. The sampled candidate set is distinct for GCD, PICORV32, BLAKE2S, IBEX and POLY1305. For every sampled canonical function, the library configuration includes all available drive-strength variants (X1 and X2), so that the SP&R tool can select among them during technology mapping.

Synthesis output and training label. After assembling each library configuration, we perform a full SP&R run, and the resulting post-route netlist provides the instance count of each cell for that configuration. For an ordered pair (A, B), the graph input contains cells used in A or legal in B, with design-context features taken only from A. The training label for each legal cell in B is its instance count in B, normalized by the total logic-cell count of A.

4 Experimental Setup

We evaluate the framework on five designs (AES, JPEG, ETH, SHA256, and KECCAK) under the SO3 PDK against three claims.[3] First, the design-specific 3-input and 4-input selections achieve greater chip-area reduction than the prior ≤3-input library and are competitive with the exhaustive same-arity library (Full 3-in / Full 3-in+4-in), while timing and DRC remain within the validity budget. Second, the selected C^{k_2} differs across designs, and substituting another design's selection degrades PPA. Third, the selected library $L_0 \cup C^{k_2}$ remains stable under moderate per-cell area estimation noise, indicating that the estimator precision is sufficient for reliable candidate selection.

Technology and library settings. All experiments use the SO3 PDK, and additional cells for expanded libraries L_i are generated using the SO3 cell-generation framework [15]. The experimental baseline L_0 is the minimal 14-cell set required for SP&R, consisting of NAND2_X1, NOR2_X1, all available drive-strength variants of INV and BUF, and flip- flop and latch cells. We use the SO2-PR mode, which fixes the circuit topology, and replace the original simultaneous placement-and-routing optimization with a faster sequential flow[4] that performs transistor placement first and internal cell routing afterward.

Benchmark set. Training-data generation uses GCD, PICORV32, BLAKE2S, IBEX and POLY1305. End-to-end evaluation uses AES, JPEG, ETH, SHA256 and KECCAK, covering both control-flow and data-flow design styles. Additional details on the benchmarks are provided in the Appendix B.

Initial libraries and key parameters. Unless stated otherwise, $k_1 = 10$, $k_2 = 5$, and $m = 3$, with one fixed mining-constraint set η, are used across benchmarks. Because k_1 counts logic functions rather than cells, pairing each function with both X1 and X2 drive-strength variants yields $2k_1 = 20$ candidate cells per design. η bounds logic-cluster cell count, input/output count, and logic depth so that the mined logic clusters are frequent enough for training and small enough for schematic generation, matching and Fusion Compiler (FC) runs. On GCD (860 instances) mining and scoring take 0.8 s and 0.5 s respectively, with cell-area computation under 0.1 s. On PICORV32 (26,598 instances), mining and scoring take 26 s and 12.4 s respectively, with cell-area computation at 0.6 s. Our framework source code is released at [26].

Design-specific library composition and comparisons. Our goal is to construct a design-specific library for each benchmark. We therefore define candidate spaces using 3-input and 4-input P-classes, i.e., logic functions realizable with up to 3 or 4 inputs. We exclude ≥ 5-input spaces because their design spaces are too large to practically cover. For each benchmark, we evaluate six library configurations: (1) a prior ≤3-input library [15], (2) the full 3-input library [27], (3) the full 3-input library augmented with prior 4-input cells [15, 27], (4) the baseline L_0, (5) Ours-3in: our design-specific 3-input selection $L_0 \cup C^{k_2}_{\text{3in}}$, and (6) Ours-4in: our design-specific 4-input selection $L_0 \cup C^{k_2}_{\text{4in}}$.

Block-level PPA evaluation protocol. For each benchmark, we evaluate block-level PPA for all six library configurations. We apply uniform PASS criterion: total DRC count ≤ 200, short-net DRC count ≤ 20, and worst negative slack (WNS) ≤ 20 ps. The TCP/ECP sweep procedure used to obtain same-area power and timing outcomes is detailed in the Artifact Appendix. Because FC results are subject to randomness, each PPA point is evaluated with three repeated runs under identical settings, and only PASS designs are used.

5 Results and Discussion

The evaluation tests whether the design-specific libraries selected by GAPMA recover the chip-area reduction of much larger exhaustive libraries while preserving the candidate-ranking accuracy of the underlying scoring components.

5.1 Ranking and Prediction Accuracy

We validate the two scoring components that drive candidate selection before the full library comparison.

Effect of PM-NPN Aggregation. Both configurations select the top k_2 candidates from the same mining output and add them to L_0 for a single SP&R run. The only difference is the ranking score: without PM-NPN aggregation, candidates are ranked by the raw count-area score $\sum_c \text{count}(c) \cdot \hat{a}_c$; with PM-NPN aggregation, the count $\text{count}(c)$ is replaced by the PM-NPN aggregated mining count before computing the score.

Table 5 reports the ratio of post-route PPA with PM-NPN aggregation to that without, across the five evaluated designs. PM-NPN aggregation reduces area by 6.3% on average (range 0 to 13.6%) and reduces ECP by 6.3% on average (range 1.5 to 18.4%), while power changes by −2.2% on average (range −6.9% to +1.8%). This ratio is

[3] Hardware and tool versions are detailed in the Artifact Appendix.

[4] Implementation details (stage timeouts, dummy-poly retry, double-height promotion) are documented in our released code [26].

Table 5: Effect of PM-NPN aggregation on post-route PPA. Each entry is the ratio (with PM-NPN) / (without PM-NPN); values < 1 indicate improvement for all three metrics.

Metric	AES	JPEG	SHA256	KECCAK	ETH	Mean
Power	**0.931**	1.018	1.013	**0.986**	**0.941**	**0.978**
ECP	**0.972**	**0.816**	**0.962**	**0.985**	**0.949**	**0.937**
Area	**1.000**	**0.911**	**0.941**	**0.968**	**0.864**	**0.937**

Table 6: Per-design held-out MAE (pp) of the GATv2 cell-usage-share predictor under LODOCV. The mixed-design-split MAE on a 20% held-out set (all five designs mixed) is 1.70 pp.

Held-out	BLAKE2S	GCD	IBEX	PICORV32	POLY1305	LODOCV mean
MAE (pp)	1.43	2.14	2.03	1.71	1.94	1.85

measured against the same pipeline without PM-NPN aggregation, whereas the reductions in Table 7 are measured against L_0.

GATv2 Cell-Usage-Share Predictor Accuracy. The GATv2 cell-usage-share predictor is evaluated under two protocols: (i) leave-one-design-out cross-validation (LODOCV), which measures cross-design generalization, and (ii) a mixed-design split, in which all five designs are mixed and a 20% held-out evaluation set is reserved, measuring within-training-distribution accuracy. Under LODOCV (Table 6), the mean absolute error (MAE), averaged across the five held-out designs, is 1.85 percentage points (pp), with per-design MAE ranging from 1.43 to 2.14 pp. Under the mixed-design split, the held-out MAE under the mixed-design split is 1.70 pp.[5]

5.2 End-to-End Evaluation

Selection quality is measured by the realized implementation result after the selected cells are added to the baseline library. The end-to-end comparison measures the selected libraries against prior and exhaustive libraries, and cross-design reuse tests whether a cell set selected for one benchmark transfers to another.

Full-Library Comparison. The selected libraries are intended to recover the area reduction of much larger libraries while adding only a few cells. Each benchmark is implemented under the six library configurations, covering both 3-input and 4-input arity targets.

Design-Specific Selection. A per-design selection is useful only when the selected cells encode design-specific cell usage rather than a universal best-five list. With $k_2 = 5$, each benchmark is implemented under its own selection, under each other benchmark's selection, and under the union of all five selections. Table 8 summarizes cross-design reuse. Diagonal entries having the largest chip-area reductions indicate design-specificity. A union column close to the diagonal indicates that the per-design selections do not conflict. The selected logic functions differ across designs because the GATv2 predictor conditions cell usage share on the design's baseline implementation and on the competing cells in the candidate pool.

Each candidate function contributes two nodes (X1 and X2) to the GATv2 inference graph, and its predicted usage share is the sum of both. The area-gain-weighted score (Equation 2) multiplies this share by area reduction, because a frequently predicted cell may target a function that L_0 already implements compactly, yielding little gain despite high usage. Cells with lower usage but higher area reduction consolidate larger existing logic clusters and dominate the score.

5.3 Pre-Layout Cell Area Estimation Accuracy

The selection objective uses the absolute pre-layout poly count (Equation 2), so the estimator must be accurate in absolute poly count, not only in rank. We evaluate 106 single-height combinational cells with completed SO3 cell layouts (44 SO3-Cell functions [15] and 62 4-input NPN-canonical cells). Comparing CDL-only poly-count estimate vs. the GDS-derived poly count gives mean absolute percentage error (MAPE) = 1.0% overall (SO3-Cell 0.0%, 4-input 1.8%). To probe selection sensitivity to estimator error, we perturb each $\hat{a}_i$ by uniform noise of ±10%, ±20%, or ±30% and rerun selection (N=50 seeds per band per design, averaged across the five evaluation designs). We report cell overlap (fraction of the k_2 noise-free cells reappearing) and rank shift (fraction of the k_2 rank slots whose occupant changes, e.g., a cell ranked 3rd dropping to 8th). Overlap is 91%/81%/74% and rank shift is 25%/50%/64% across the three bands. See Appendix C.

[5]Because each label is a per-cell cell usage share that sums to one across the library, a mean MAE of 1.85 pp corresponds to, for example, predicting NAND2 at 28.15% when its realized cell usage share is 30%.

Table 7: End-to-end comparison across six library configurations. "norm." is normalized relative to *Prior-3in*; ECP: effective clock period. *Ours-3in*/*Ours-4in*: cell-usage-share-guided selections adding k_2 cells to L_0.

			Power		ECP		Area	
Design	Library	#cells	(mW)	norm.	(ns)	norm.	(μm^2)	norm.
AES	Prior-3in	35	2.61	1.000	0.130	1.000	352.97	1.000
	Full-3in	99	2.60	0.997	0.131	1.008	342.56	0.971
	Full-3in+4in	117	2.60	0.994	0.132	1.015	342.56	0.971
	L_0	14	2.80	1.071	0.146	1.126	433.90	1.229
	Ours-3in	31	2.54	0.972	0.124	0.956	342.56	0.971
	Ours-4in	31	2.44	0.934	0.130	1.001	360.77	1.022
JPEG	Prior-3in	35	2.48	1.000	0.160	1.000	1443.39	1.000
	Full-3in	99	2.63	1.059	0.161	1.006	1396.36	0.967
	Full-3in+4in	117	2.57	1.035	0.152	0.950	1354.97	0.939
	L_0	14	2.69	1.083	0.209	1.307	1788.52	1.239
	Ours-3in	31	2.52	1.017	0.145	0.906	1422.37	0.985
	Ours-4in	31	2.46	0.992	0.158	0.987	1365.26	0.946
ETH	Prior-3in	35	7.45	1.000	0.128	1.000	2027.83	1.000
	Full-3in	99	7.41	0.994	0.121	0.945	1865.83	0.920
	Full-3in+4in	117	7.42	0.995	0.127	0.992	1788.52	0.882
	L_0	14	7.73	1.037	0.139	1.089	2311.05	1.140
	Ours-3in	30	7.32	0.983	0.139	1.086	1875.96	0.925
	Ours-4in	29	7.26	0.975	0.130	1.013	1898.25	0.936
SHA256	Prior-3in	35	1.61	1.000	0.154	1.000	514.78	1.000
	Full-3in	99	1.64	1.018	0.155	1.006	523.18	1.016
	Full-3in+4in	117	1.67	1.035	0.155	1.006	494.97	0.962
	L_0	14	1.67	1.034	0.192	1.248	596.84	1.159
	Ours-3in	29	1.59	0.986	0.157	1.019	514.78	1.000
	Ours-4in	29	1.68	1.042	0.176	1.141	489.77	0.951
KECCAK	Prior-3in	35	1.25	1.000	0.160	1.000	604.78	1.000
	Full-3in	99	1.26	1.015	0.141	0.884	584.37	0.966
	Full-3in+4in	117	1.24	0.998	0.139	0.871	575.49	0.952
	L_0	14	1.35	1.088	0.175	1.093	860.30	1.423
	Ours-3in	29	1.26	1.009	0.159	0.993	604.78	1.000
	Ours-4in	29	1.24	0.993	0.136	0.848	569.88	0.942

Table 8: Cross-design candidate-cell reuse area ratio relative to self, *Ours-3in*. Rows = implemented design; columns = source design for selection (or union of all five). Final row = number of added candidate cells.

	Candidate cells from					
Design	AES	JPEG	ETH	SHA256	KECCAK	Union
AES	1.000	1.044	1.058	1.044	1.136	0.986
JPEG	1.065	1.000	1.065	1.000	1.065	1.065
ETH	1.109	1.000	1.000	1.000	1.000	1.000
SHA256	0.978	1.000	1.000	1.000	1.000	0.962
KECCAK	1.095	1.048	1.095	1.095	1.000	1.095
# of cells	31	31	30	29	29	47

6 Conclusion

We present GAPMA, a framework for design-specific standard-cell library composition that performs drive-strength-aware, competition-aware, learning-based candidate selection without requiring candidate cell layout generation. GAPMA aggregates mining counts across polarity variants of each NPN class, and bases candidate-cell selection on GATv2-based joint cell-usage-share prediction over proposed library augmentations. GAPMA's design-specific composition improves post-route implementation QoR over baseline and prior libraries: SP&R runs using k_2=5 added cells reduce chip area by 14–30% relative to L_0. Ongoing work focuses on scalability to 5-input and larger candidate spaces, along with improved CDL generation and cell synthesis [28], function coverage and generalization.

Acknowledgments

This work is partially supported by the Samsung AI Center.

References

[1] K. Scott and K. Keutzer, "Improving Cell Libraries for Synthesis," in *Proc. IEEE Custom Integrated Circuits Conference (CICC)*, 1994, pp. 128-131.

[2] H. Yoshida, "Optimal Generation of Design-Specific Cell Libraries," Ph.D. dissertation, Univ. of Tokyo, 2007.

[3] M. Rahman, R. Afonso, H. Tennakoon and C. Sechen, "Power Reduction via Separate Synthesis and Physical Libraries," in *Proc. ACM/IEEE Design Automation Conference (DAC)*, 2011.

[4] P. Majumder, B. Kumthekar, N. R. Shah, J. Mowchenko, P. A. Chavda, Y. Kojima, H. Yoshida and V. Boppana, "Method of IC Design Optimization via Creation of Design-specific Cells from Post-layout Patterns," U.S. Patent App. US 2008/0127000 A1, 2008.

[5] D. Motiani, V. Kheterpal and L. T. Pileggi, "Method for the Definition of a Library of Application-domain-specific Logic Cells," U.S. Patent App. US 2013/0007676 A1, 2013.

[6] C. Pilato, F. Ferrandi and D. Pandini, "A Fast Heuristic for Extending Standard Cell Libraries with Regular Macro Cells," in *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2010, pp. 23-28.

[7] C.-K. Cheng, B. Kang, B. Lin and Y. Wang, "Standard Cell Layout Generation: Review, Challenges, and Future Works," in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2025.

[8] M. Lefebvre, D. Marple and C. Sechen, "The Future of Custom Cell Generation in Physical Synthesis," in *Proc. ACM/IEEE Design Automation Conference (DAC)*, 1997, pp. 446-451.

[9] V. K. Salimath and C. Sechen, "Essential Standard Cell Library Composition," in *Proc. IEEE Dallas Circuits and Systems Conference (DCAS)*, 2022, pp. 1-6.

[10] B.-S. Kim, H.-S. Won, T. H. Han and J.-S. Yang, "AFSEM: Advanced Frequent Subcircuit Extraction Method by Graph Mining Approach for Optimized Cell Library Developments," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 662-665.

[11] T. Liang, J. Chen, L. Li and W. Zhang, "AutoCellLibX: Automated Standard Cell Library Extension Based on Pattern Mining," *arXiv:2207.12314*, 2022.

[12] R. Fu, C. Wang, B. Yu and T.-Y. Ho, "TeMACLE: A Technology Mapping-Aware Area-Efficient Standard Cell Library Extension Framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 44, no. 8, pp. 3034-3045, 2025.

[13] J. Y.-C. Lee, Y.-J. Su, J.-C. Tsai, A. C.-W. Liang, C. H.-P. Wen and H.-M. Huang, "SOFA-H: Post-Synthesis Area Optimization via Functionally Encoded, Net-Driven Subgraph Mining and SAT-Based Hypercell Remapping," in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2026, pp. 800-806.

[14] Y. Ren, Y. Wang, X. Meng, G. Cheng, B. Peng, L. Zhang, Y. Lin, R. Wang and G. Sun, "CellE: Automated Standard Cell Library Extension via Equality Saturation," in *Proc. ACM/IEEE Design Automation Conference (DAC)*, 2026.

[15] C.-K. Cheng, A. B. Kahng, B. Kang, S. Kang, J. Lee and B. Lin, "SO3-Cell: Standard Cell Layout Automation Framework for Simultaneous Optimization of Topology, Placement, and Routing," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025, pp. 1-9. https://github.com/ckchengucsd/SO3-Cell

[16] L. Benini and G. De Micheli, "A Survey of Boolean Matching Techniques for Library Binding," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 2, no. 3, pp. 193-226, 1997.

[17] D. Bhattacharya, V. Boppana, R. Murgai and R. Roy, "Process for Automated Generation of Design-Specific Complex Functional Blocks to Improve Quality of Synthesized Digital Integrated Circuits in CMOS Using Altering Process," U.S. Patent 7,003,738 B2, 2006.

[18] Y. Ren, B. Peng, C. Xue, K. Guo, Y. Wang, G. Cheng, Y. Lin, L. Zhang and G. Sun, "Orthrus: Dual-Loop Automated Framework for System-Technology Co-Optimization," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025.

[19] D. T. H. Lyn, A. bin Marzuki and J. L. M. May, "A Methodology for Cell Merging Circuit Transformation on Post-placement High Speed Design," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 2, no. 1, pp. 1-6, 2012.

[20] J. Yoon, J. Jeong and H. Park, "Breaking Standard Cell Margin Constraints for Area-Efficient VLSI Design," in *Proc. Design, Automation and Test in Europe Conference (DATE)*, 2026.

[21] X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining," in *Proc. IEEE International Conference on Data Mining (ICDM)*, 2002, pp. 721-724.

[22] T. Uehara and W. M. vanCleemput, "Optimal Layout of CMOS Functional Arrays," *IEEE Transactions on Computers*, vol. C-30, no. 5, pp. 305-312, 1981.

[23] R. L. Maziasz and J. P. Hayes, "Layout Optimization of Static CMOS Functional Cells," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 9, no. 7, pp. 708-719, 1990.

[24] S. Brody, U. Alon and E. Yahav, "How Attentive are Graph Attention Networks?" in *Proc. International Conference on Learning Representations (ICLR)*, 2022.

[25] K. T. Talluri and G. J. van Ryzin, *The Theory and Practice of Revenue Management*. Boston, MA: Springer, 2004.

[26] GAPMA GitHub repository for GAPMA. https://github.com/ABKGroup/GAPMA.

[27] B. Kang, A. Mishchenko, M. Fujita, B. Lin and C.-K. Cheng, "L2L: Logic to Layout Exploration of Standard Cell Library Design," in *Proc. ACM/IEEE Design Automation Conference (DAC)*, 2026. https://github.com/b8kang/L2L-Logic_to_Layout_Exploration

[28] M. S. Cardoso, G. H. Smaniotto, A. A. O. Bubolz, M. T. Moreira, L. S. da Rosa Jr. and F. S. Marques, "Libra: An Automatic Design Methodology for CMOS Complex Gates," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 10, pp. 1345-1349, 2018.

Appendices

A GATv2 Feature Groups, in Detail

Table 4 in the main text lists the 46-dimensional node vector and the 7-dimensional edge vector. This appendix gives a short prose explanation of each feature group: what the features measure; what makes them PDK-, function-, or design-derived; and why each feature group is included.

A.1 Node features (46)

CDL (9). The CDL features read the cell's schematic before layout: input pin count, NMOS and PMOS transistor counts, NMOS and PMOS total fin counts and their ratio, the longest source–drain series stack in each network, and the pre-layout poly count of Equation 1. The ratio and stack-depth entries are kept with the raw counts so the network does not have to learn them, and the poly count value is reused as the area objective in Equation 2. This feature group is PDK-aware through the NFin parameter but design-agnostic — the same cell produces the same CDL vector for any design.

Truth-table (9). The truth-table features describe Boolean properties of the function that the cell computes: Hamming weight of the truth table (the number of input assignments mapping to output 1), algebraic-normal-form (ANF) degree, affine / symmetric / monotone / anti-monotone flags, the input- and output-inverter counts measured at the PM-NPN representative, and total transistor count. This feature group is PDK- and design-agnostic; two cells that compute the same function on different PDKs share the same truth-table vector. This lets the GATv2 share information between cells with similar functions even when their CMOS implementation differs.

Design (21). The design features describe how the cell is used in the current implementation A: a membership flag, the cell's share of all instances, fanout statistics, timing diagnostics (violation ratio, path coverage, worst-slack rank), per-pin electrical statistics (output capacitance, input slew, output slew), dynamic-power statistics, and the cell's rank in the mining-frequency table. Timing statistics are aggregated over many critical paths so that they capture the cell's role across the chip rather than in specific paths; dynamic power uses a uniform 0.1 toggle rate so that the feature captures both design and library properties.

Library role (3). Three one-hot flags indicate where the cell is in the current sample $(A, B_i(S))$: baseline-only when the cell is in L_0 but not in the candidate set; shared when it is in both; and candidate-only when it is in the candidate set only. The library-role feature group is the only one that depends on S, so these are the only feature that changes when the GATv2 is asked to score different candidate subsets.

NPN rep. (4). Four features describe the cell's place in the NPN equivalence-class system: a flag for the PM-NPN representative; the number of input inverters removed when the cell is rewritten as its PM-NPN representative; a flag for output-inversion removal; and the NPN-class size (number of P-canonical clusters mapping to the same NPN class as this cell). The size feature is a proxy for how widely useful a candidate cell is across the design: — a small class is implemented by one or two clusters, while a large class can absorb usage from many sites.

A.2 Edge features (7)

Truth-table (4). These edge features tell the network whether the two cells are interchangeable: a same-truth-table flag, a same-NPN-class flag, and the input- and output-inverter-count differences between the two cells. The two flags carry the function-level interchangeability signal: the two diffs let the network learn that adding an inverter costs much less than changing logic functions.

Mining (3). These edge features add the design-specific signal that the function- and PDK-derived features do not provide: a co-occurrence ratio (mined clusters containing both cells, normalized by the smaller cell's total cluster count), the ratio of mining counts between the two cells, and the cluster-footprint overlap as a fraction of the union.

B GATv2 Training Details

This appendix records the architecture, loss, optimization, and per-design compute budget behind the GATv2 model of Section 3.5.

Architecture. The network has three GATv2 convolutional layers (width 128, four attention heads, dropout 0.3, exponential linear unit (ELU) between layers).

Loss. The loss sums the cross-entropy on the distribution head, the mean squared error (MSE) on the scale head (equal weight), and a pairwise margin loss (margin = 0.02, weight 0.1) on cell rank.

Optimization. Adam (learning rate 8×10^{-4}, weight decay 10^{-3}) with linear warmup over 8 epochs, batch size 128.

Data and compute. Per design, the SP&R run counts are BLAKE2S (102), GCD (209), IBEX (105), PICORV32 (146), and POLY1305 (79), yielding 92K ordered training pairs in total. Training takes approximately one hour on a single NVIDIA A100 GPU.

C Cell-Level Area Prediction

The scoring stage needs an area estimate per candidate cell. Measuring it directly requires a full PDK characterization (layout, extraction, Liberty generation), which is costly at hundreds of candidates per design. Therefore, the GAPMA pipeline uses a cell-level area predictor that turns the cell's CDL transistor list into a single area number: this same number feeds the area-gain selector of Section 3.2.

Inputs. For each cell the predictor reads the device list emitted by the CDL parser: device count by polarity; fin count per transistor (or implicit fin count when the CDL omits the parameter); and the minimum poly count derived from the standard cell row height.

Model form. The predictor sums per-device contributions: NMOS and PMOS contribute proportionally to their fin counts, with a PDK-specific per-fin width w_n, w_p: fixed overhead (well taps, end caps) is folded into an additive PDK constant.

Calibration. Coefficients are fit once per PDK by least-squares against the cell areas of all L_0 baseline cells. We use the residual distribution of this fit as the area-prediction uncertainty supplied to the downstream selection.

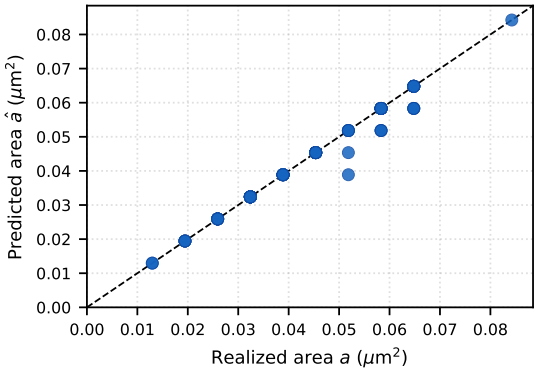

Figure 5: Predicted ($\hat{a}$) versus realized (a) cell area for the 106 single-height combinational cells used to validate the predictor (44 SO3-Cell baseline functions and 62 4-input NPN-canonical cells). The dashed diagonal is exact prediction. Overall MAPE is 1.0% (SO3-Cell 0.0%, 4-input 1.8%).

D Artifact Appendix

D.1 Abstract

The artifact is the training-data generation, GATv2 training, and iterative candidate-cell selection pipeline of this paper, packaged as a code repository with all RTL benchmarks, an academic 3 nm PDK, and a pretrained model. Three top-level scripts reproduce the pipeline end to end. Synthesis and P&R need a commercial license. The mining and GATv2 code paths run standalone in Python.

D.2 Artifact Meta-information Checklist

- **Algorithm:** A pipeline consisting of PM-NPN mining-count aggregation; pre-layout area estimation via a common Euler path; GATv2 joint cell-usage-share prediction; ratio-threshold cell-count resolution; and ILP-based cell layout generation that performs transistor placement followed by in-cell routing.
- **Program:** The 10 benchmarks of Section 4, included under `rtl/`.
- **EDA Tools:** Synopsys Fusion Compiler T-2022.03-SP2 for synthesis and P&R; Gurobi 11.0.1 for standard cell layout generation; Cadence Pegasus v21.3, Quantus v21.1, and Liberate v23.2.3 for cell validation and enablement generation; and Synopsys Library Compiler L-2016.06-SP3-1 for converting the generated Liberty files to DB files. The commercial tools are not included; only the scripts used to run them are provided.
- **Compilation:** CMake and a C++17 compiler for the mining toolchain, which links the Yosys submodule.
- **Model:** Pretrained GATv2 model at `data/GATv2/model_pretrained.pt`, with normalization statistics embedded.
- **Data set:** RTL, the PDK of 369 cells (312 of them being mined custom cells), and the shared cell-feature and mining relation table. The per-design raw training data is not included, but can be regenerated using `flow/datagen.sh`.
- **Run-time environment:** Linux x86_64, Python 3.9 with PyTorch, PyTorch Geometric, and Gurobi.
- **Hardware:** Two Intel Xeon Gold 6148 CPUs with 40 cores and 80 threads total, and 192 GB of main memory. GATv2 training uses a single NVIDIA A100, as detailed in Appendix B.
- **Metrics:** Post-route chip area and power satisfying the PASS criteria of Section 4. Training reports the MAE for cell-usage-share prediction on held-out designs, Table 6.
- **Execution:** Mining and scoring take 0.8 s and 0.5 s on the 860-instance GCD design and 26 s and 12.4 s on the 26,598-instance PICORV32 design. Every PPA point is a full Fusion Compiler SP&R run, repeated three times.
- **Disk space required:** The repository is approximately 1.5 GB, including third-party submodules (Yosys, SO3-Cell, OpenDB). Per-design training and experimental data are not included.
- **Public availability:** Public GitHub repository, with an archived snapshot on Zenodo.
- **Code licenses:** BSD 3-Clause.
- **Zenodo DOI:** `10.5281/zenodo.21987869`

D.3 Description

D.3.1 How to access. The public repository is at https://github.com/ABKGroup/GAPMA. The archived snapshot is at https://doi.org/10.5281/zenodo.21987869, which identifies the archived snapshot used for artifact evaluation.

D.3.2 Software. Python 3.9 with PyTorch, PyTorch Geometric, and NumPy, Gurobi and CMake with a C++17 compiler. Cadence Pegasus, Quantus, and Liberate and Synopsys Library Compiler are additionally required when a selection picks a cell that the included PDK does not already provide.

D.4 Installation

(1) Clone the repository and `cd GAPMA`. The three submodules are empty directories, with their upstream URLs in `.gitmodules`.
(2) Apply `tools/cell_generation/scripts/so3-cell.patch` inside the `SO3-Cell` checkout.
(3) Set `PDK_ROOT` to the repository's own `pdk/SO3`, then set `FC_BIN`, `FC_LICENSE`, `GUROBI_HOME`, and `YOSYS_ROOT`.
(4) `make -C tools/third_party/yosys`, then `tools/mining/build.sh`.

D.5 Evaluation and Expected Results

`flow/autorun.sh --config flow/config/autorun.conf` on one design/arity combination runs the loop from the D_0 baseline to D_3: baseline synthesis and P&R, mining, GATv2 prediction, candidate selection, generation of any selected cell absent from the PDK, and a final SP&R run with the augmented library. Per-combination driver logs land in `<fc-root>/autorun_logs/`, and each SP&R run writes a `qor.csv` in its own run directory.

Measurement protocol. To measure area and power, we first sweep the target clock period (TCP) and select the largest feasible target whose achieved effective clock period (ECP) is within 5 ps of TCP. We then fix that TCP and reduce the area to find the smallest-area PASS implementation; the resulting area is reported as the library block area, and the minimum power observed at that area is reported as the library block power. To compare timing at the same area, we further sweep the area and TCP and identify, for each library, the point beyond which additional area no longer improves ECP, then compare how much ECP each library can reduce under the same area condition.

The expected end state is the L_0 and *Ours-3in*/*Ours-4in* rows of Table 7. As stated in Section 4, Fusion Compiler results exhibit random variation, so each PPA point is evaluated with three repeated runs under identical settings and only PASS designs are used.