Gear-Ratio-Aware Standard Cell Layout Framework for DTCO Exploration

Chung-Kuan Cheng, Andrew B. Kahng, Bill Lin, Yucheng Wang*, Dooseok Yoon
University of California, San Diego
La Jolla, California, USA
{ckcheng,abk}@ucsd.edu,billlin@eng.ucsd.edu,{yuw132,d3yoon}@ucsd.edu

Abstract

With advances in lithography technology, the minimum metal pitch (MP) becomes smaller than the contacted poly pitch (CPP). This difference has long prompted the need to seek an optimal ratio between CPP and MP. Automated cell synthesis with conditional design rules offers a valuable lever to speed up the technology exploration process and to identify the best "gear ratio" (GR) for Design-Technology Co-optimization (DTCO) exploration.

Existing approaches for cell layout generation frameworks have primarily supported uniform grids with limited gear ratio options. In this work, we present SMTCell, a new exploratory framework for cell layout generation that allows flexible gear ratio options using a graph-based data structure. We employ distance-based objective functions and conditional design rule parameters to adapt to varying pitch values. This approach enables us to investigate and discover optimal layouts under diverse technology node settings. An acceleration feature drastically trims the solution space, resulting in a speed increase of up to 19× without sacrificing the quality of the original solutions.

With SMTCell, cell synthesis automation can be configured to accommodate a wide range of design choices. We conduct an empirical study to assess the impact of gear ratio at block-level synthesis, place and route (SP&R) outcomes, with the ultimate goal of identifying the most effective technology and standard cell configurations in terms of design power, performance and area (PPA) metrics.

CCS Concepts: • **Hardware → standard cell libraries**.

Keywords: standard cell layout, Satisfiability modulo theories, Design technology co-optimization

* Corresponding author.

SLIP '23, November 2, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0474-1/23/11.
https://doi.org/10.1145/3632409.3640475

ACM Reference Format:
Chung-Kuan Cheng, Andrew B. Kahng, Bill Lin, Yucheng Wang*, Dooseok Yoon. 2023. Gear-Ratio-Aware Standard Cell Layout Framework for DTCO Exploration. In *2023 ACM International Workshop on System-Level Interconnect Pathfinding (SLIP '23), November 2, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3632409.3640475

1 Introduction

The *gear ratio* (GR) [4] between *contacted poly pitch* (CPP)[1] and *M1* pitch[2] plays a pivotal role in overcoming scaling constraints, as it enables fine-grain balancing of density, pin access and routability considerations. To support Design-Technology Co-optimization (DTCO) [13] and exploration of potential metal pitch and offset[3] [6] [18] configurations, we develop an improved Satisfiability Modulo Theories-based *cell layout synthesis* solver, called *SMTCell*, that accommodates various GR configurations. *SMTCell* leverages a graph-based methodology to handle rich configurations of pitch and design rules while returning optimal cell layouts.

Motivations for such a tool are seen as follows.

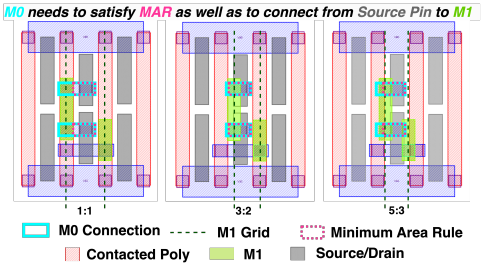

Figure 1. INV_X2 layouts with 1:1 GR (left), 3:2 GR (middle) and 5:3 GR (right). M0 connection requires expansion to satisfy the Minimum Area Rule (MAR) and to make via interconnections.

Gear ratio has a profound impact on the allocation of routing resources across various layers.[4] Figure 1 shows three

[1] Distance between a poly gate and its adjacent poly gate.
[2] Distance between adjacent tracks on the lowest vertical routing layer.
[3] Offset is the distance by which all vertical routing tracks move to the right from the left edge of the cell.
[4] In this work, we assume a layer stack of placement (vertical), metal 0 (M0, horizontal), metal 1 (M1, vertical) and metal 2 (M2, horizontal). "Placement" in our work denotes a layer that contains gate, source and drain. We assume that *M1* and *M2* are used for input/output (IO) connections. In the following, we refer to these four layers (placement through *M2*) respectively as L^1 through L^4.

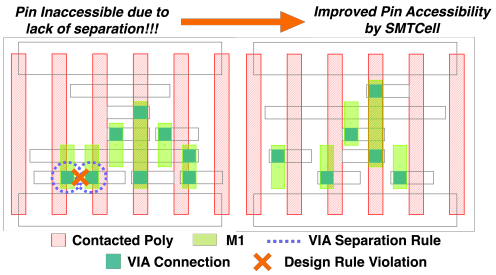

Figure 2. AOI22_X1 layout: pin unroutable due to via rule violation (left) and improved pin accessibility (right). Unroutability is caused by the previous design rule formulation failing to consider the varying pitches induced by GR.

INV_X2 layouts under different GR settings. As *M1* pitch decreases (going from left to right in the figure), we observe varying degrees of utilization in *M0* routing resources. Shapes on *M0* may be stretched to satisfy the Minimum Area Rule (MAR) [17]. And, to ensure proper interconnectivity, sufficient metal length must be provided to facilitate connections between upper and lower vias. The positioning of these vias is influenced by the varying pitch value on *M1*. Notably, in larger cells where routing resources tend to be scarce, efficient routing across different layers becomes challenging. This challenge is further exacerbated when factoring in the Via-separation Rule (VR) [17] and other layout constraints.

Furthermore, imposition of a more stringent *M1* pitch can introduce more complexity to layout design. Pin openings on adjacent tracks become unroutable from *M2*, as highlighted in orange in Figure 2. A layout automation tool must be precise enough to take advantage of such dense routing tracks while being aware of potential spacing violations caused by VR and other design rules.

Previous methods [9] [14] demonstrate the potential benefit of using SMT-based design rule constraints to simultaneously place and route when generating cell layouts. SP&R [9] uses a grid graph representation of a potential cell layout, then encodes graph elements into boolean variables. Cheng et al. [2] [3] leverage this approach on newer device architectures (Vertical-FET [11] and Complementary-FET [15]), and demonstrate scaling boost at block-level. Recent enhancements of the PROBE framework [5] automate custom Process Design Kit (PDK) generation and incorporate PPA and IR drop prediction for DTCO.

However, previous formulations are limited to only handling 1:1 and 3:2 GR. Overcoming this limitation requires a new and effective way to represent the solution space with a graph structure. Such a graph structure needs to encode the varying lengths of edges (i.e., metal segments). Specifically, to enable all (spacing-related) design rule constraints[5] to be satisfied, preference should be given to shorter metal segments.

In this work, we incorporate a new graph structure with properly scaled edges to achieve a SMT-based cell layout synthesis. We present *SMTCell*, a comprehensive framework endowed with an array of design options. Introduction of the underlying graph structure enables incorporation of dense routing tracks into cell layout, and exploration of arbitrary gear ratios. Furthermore, we introduce *pre-partitioning*, a solution space trimming technique, to shorten runtime.

Our main contributions are as follows:

- We propose a new graph structure, called the *relative layered grid graph*,[6] which supports all gear ratios by representing densely placed routing tracks on each layer. Our formulation based on this graph enables distance-based objective functions and design rule constraints.
- We introduce a new cell partitioning scheme that "hints" a more plausible SMT solution space, thereby speeding up runtime by up to 19×.
- We use our framework to explore different GR options and the impact of GR on block-level Power, Performance and Area (PPA).

SMTCell is available in permissive open source, in our GitHub repository [19]. In the following, Section 2 introduces some basic concepts and a graph structure to explore gear ratio. Section 3 introduces a solution space reduction method with cell partitioning and cell width hint. Finally, Section 4 uses *SMTCell* to conduct assessments with both cell-level and block-level metrics.

2 Formulation

Recall that *gear ratio* (GR) is the ratio between (contacted) *Poly* and *M1* pitches. GR is typically given in integers or half-integers. For example, designers may use 3:2 or 2:1 GR.

Given any GR configuration, our tool dynamically formulates SMT constraints based on a graph structure. In terms of SMT formulation, we incorporate largely the same set of constraints as SP&R [9]. The novelty of *SMTCell* lies in its more fine-grained graph structure that permits capture of all the gear ratio details, which are essential for the SMT formulation. Our framework treats pitches as variables to construct a relative layered grid graph,[7] representing a potential cell layout.

In our graph structure, we typically consider a 4-layer design: placement layer L^1 (vertical), *M0* layer L^2 (horizontal), *M1* layer L^3 (vertical), and *M2* layer L^4 (horizontal). Since our SMT formulation is based on constraints involving the vertices of the graph, it is crucial to consider the varying distances between vertices to accurately formulate various design rules [17].

[5] In this work, we refer to (spacing-related) design rule constraints as geometric constraints imposed on standard-cell layout designs. Design rule parameters are user-given values to control these constraints.

[6] Each layer constructs a grid relative to its adjacent layers.

[7] Each layer may have a different number of columns.

2.1 Relative Layered Grid Graph Construction

Before performing graph construction, we need to compute the total cell width, and the set of columns and rows on each layer. Acquiring these information beforehand can ease the process of graph construction. The total cell width w_{total}, is determined by $\max(w_p, w_n)$, where w_p and w_n are the respective sums of PMOS and NMOS device widths (in nm),

$$w_{\text{total}} = \max(w_p, w_n), w_p \in \mathbb{Z}^+, w_n \in \mathbb{Z}^+.$$

Let i be the index of the layers. With w_{total} and user-given variables such as a number of horizontal tracks $t \in \mathbb{Z}^+$, a set of pitch values $MP = \{mp^i \in \mathbb{Z}^+ | i \in \{1, ..., 4\}\}$ and a set of offsets from the left edge $\Delta = \{\delta^i \in \mathbb{Z}^+ | i \in \{1, 3\}\}$ of the cell,[8] we can compute a set of column sets $\mathcal{C} = \{C^i | i \in \{1, ..., 4\}, C^i \subset \mathbb{Z}^+\}$ and the a set of row sets $\mathcal{R} = \{R^i | i \in \{1, ..., 4\}, R^i \subset \mathbb{Z}^+\}$ for each layer L^i. Our targeted cell layout is uni-directional, meaning that each layer has tracks running in a direction orthogonal to that of its neighbor layer(s), with mp^i as the pitch between adjacent tracks. Therefore, a horizontal layer shares columns with its adjacent vertical layer(s); a vertical layer shares rows with its adjacent horizontal layer(s). Since gear ratio is only defined between vertical layers, horizontal layers must have the same pitch (i.e., $mp^2 = mp^4$) and thus all layers must share the same set of rows,

$$R^1 = R^2 = R^3 = R^4 = \{h \cdot mp^i | h \in \mathbb{Z}^+, h \leq t\}, i = 2 \text{ or } 4.$$

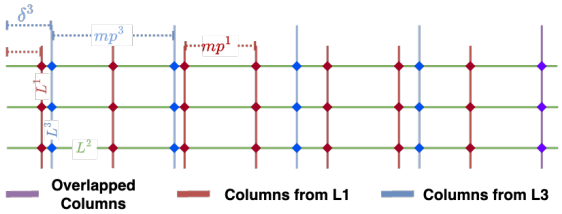

Figure 3. Top-down view of the irregular pattern of columns on L^2 induced by different pitch values on L^1 and L^3.

Creating the column sets C^i must allow for $mp^1 \neq mp^3$, and must consider offsets Δ. Figure 3 illustrates the scenario where the mp^1 and mp^3 values result in an irregular pattern of columns on L^2. The column set C^2 must contain all columns from both C^1 and C^3 to represent the densely placed routing tracks, with overlapped columns being merged to avoid redundancy. We achieve this using Algorithm 1, which iteratively creates columns for vertical layers first (Lines 1-7) and then merges them for horizontal layers without repetition (Lines 8-9), ensuring uniqueness for each column in a horizontal layer.

After creating $\mathcal{C}$ and $\mathcal{R}$, we construct a *relative layered grid graph* $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are a set of disjoint

[8] In practice, δ^1 is always set to 0 and is omitted in the calculation.

Algorithm 1: Column Set Creation

```
1  for i ∈ {1, 3} do
2      c ← δ^i
3      C^i ← ∅
4      while c ≤ w_total do
5          C^i ← C^i ∪ {c}
6          c ← c + mp^i
7      C ← C ∪ C^i
8  for i ∈ {2, 4} do
9      C^i ← C^{i−1} ∪ C^{i+1}        // Implicitly, C^5 = ∅
10     C ← C ∪ C^i
11 return C
```

vertex sets and a set of disjoint edge sets,[9] respectively:

$$\mathcal{V} = \{V^i | i \in \{1, ..., 4\}, V^i = \{v | v = (i; r; c), r \in R^i, c \in C^i\}\},$$
$$\mathcal{E} = \{E^i | i \in \{1, ..., 4\}, E^i = \{e | e = (v_1, v_2),\ v_1 \in V^i, v_2 \in V^j, j \in \{i-1, i\}\}\}.$$

Each vertex v is a triplet $(i; r; c)$ that contains layer i, row r, and column c. Each edge e is a pair of vertices whose locations are adjacent (i.e., nearest neighbors) created along the direction of this layer. To generate $\mathcal{V}$ and $\mathcal{E}$, we use Algorithm 2: for horizontal layers, edges are created along each row (Lines 8-10); for vertical layers, edges are created along each column (Lines 11-13). These edges facilitate intra-layer routing. Edges between layers, called vias, are created if and only if the two vertices (endpoints) are on adjacent layers and have the same *row* and *column* (Lines 14-17).

We use a standard multicommodity flow model for routing [7].[12] As shown in Figure 4, each multi-pin net needs to be routed from the internal pins (multiple sources) on L^1 to the frontside IO pins (multiple sinks) on L^3 and L^4. These net routings can be represented by flows (edges) through access points at each layer (vertices). *SMTCell* simultaneously performs placement and routing to ensure the optimal solution. Under our formulation, the SMT solver, *Z3* [12], minimizes a set of objective functions regarding cell metrics while satisfying the design rule constraints.

2.2 Fine-grain Design Rule Checking

SMTCell incorporates conditional design rules [17], such as End-Of-Line spacing (EOL), Step Height Rule (SHR), Parallel Run Length (PRL), Minimum Area Rule (MAR), etc. The

[9] Implicitly, V^0, C^5, etc. are empty sets, for convenience of exposition.

[10] The nesting and indexing in for loops ensures that the vertices v_a, v_b, v_c have been created before the associated edges are created. By convention, edges between layer i and layer $i-1$ are assigned to E^i.

[11] In the negative direction along the column.

[12] The multicommodity flow model optimizes the distribution of disjoint flows in a capacity-constrained network from sources to sinks.

Algorithm 2: Relative Layered Grid Graph Construction

```
for i ∈ {1...4} do
    V^i ← ∅, E^i ← ∅
    // In the order of increasing rows and columns[10]
    for r ∈ R^i do
        for c ∈ C^i do
            v ← (i; r; c)
            V^i ← V^i ∪ {v}
            if layer i is horizontal then
                Get the nearest left vertex v_a in V^i
                E^i ← E^i ∪ {{v, v_a}}
            if layer i is vertical then
                Get the nearest lower[11] vertex v_b in V^i
                E^i ← E^i ∪ {{v, v_b}}
            // if via can be constructed below
            v_c ← (i − 1; r; c)
            if v_c ∈ V^{i−1} then
                E^i ← E^i ∪ {{v, v_c}}
    V ∈ V^i, E ∈ E^i
return V, E
```

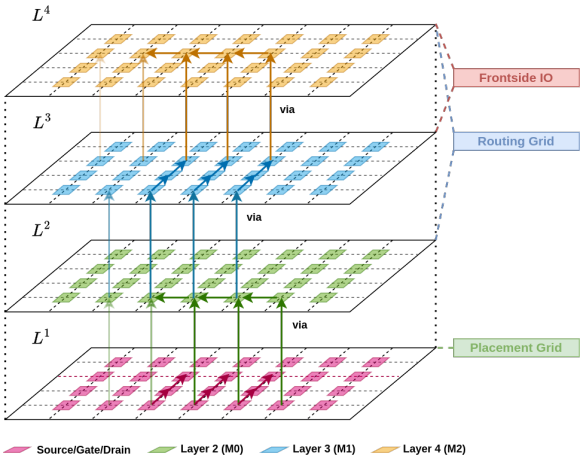

Figure 4. An example of routing from both source/drain and gate locations in the placement grid to the frontside IO through L^2, L^3 and L^4. At each routing layer, interconnections are made following the layer orientation. Between each pair of layers, vias are constructed at access points (colored squares) to make connections.

non-uniformity in a relative layered grid graph requires that the above design rule parameters be defined in terms of Manhattan distance. Prior methods [7] [9] [14] define these design rule parameters in terms of "number of vertices", which is valid only in a uniform grid where all edge lengths are equal.

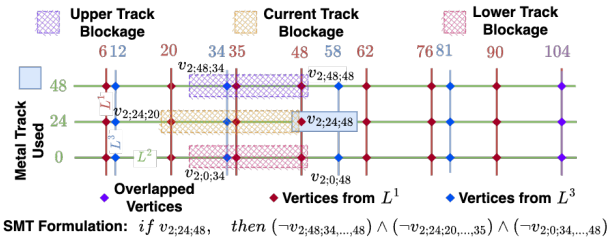

Figure 5. End-Of-Line Rule (EOL) checking on L^2 with distance input. Let $v_{2;24;48}$ denote the metal usage at layer 2, row 24 and column 48. The EOL constraint induces blockages on the current track (as $v_{2;24;20,\ldots,35}$), the upper track (as $v_{2;48;34,\ldots,48}$), and the lower track (as $v_{2;0;34,\ldots,48}$), enforcing the required separation distance.

We input design rule parameters with distance and monitor the total distance traveled along each track until the desired distance is attained. Figure 5 illustrates for a horizontal metal layer how imposing EOL at $v_{2;24;48}$ prohibits the use of vertices along three tracks: the current track (vertices $v_{2;24;20,\ldots,35}$), the upper track (vertices $v_{2;48;34,\ldots,48}$), and the lower track (vertices $v_{2;0;34,\ldots,48}$). Similarly, when applying other design rules, we examine all potential metal placements, and navigate around their neighboring vertices to prohibit access within the given distance.

2.3 Distance-based Objective Functions

To ensure the optimality of our solution, we perform a lexicographic distance-based multiple-objective optimization:

$$\text{Placement (Cell Width)} = \max(|C^i| \times mp^i + \delta^i), i \in \{1, 3\} \tag{1}$$

$$\text{Routability (\#Top Track)} = \sum_{e \in E^4} \gamma_4 \times m_e \times l(e) \tag{2}$$

$$\text{Metal Length} = \sum_{i \in \{1,\ldots,4\}} \sum_{e \in E^i} m_e \times l(e) + \gamma_{\text{via}} \times u_e, \tag{3}$$

where $l(\cdot)$ is a function that returns the edge length; m_e is an indicator for metal usage; u_e is an indicator for via usage; and γ_4 and γ_{via} are user-determined multipliers to further penalize top track usage and via usage, respectively. Equation 1 and Equation 3 minimize the resources taken by each cell, while Equation 2 ensures that the IO pins are accessible by the router.

Specifically, Equation 2 minimizes the total wirelength used on the frontside (on L^4) to prevent potential routing blockages. Equation 3 minimizes the total wirelength used. In previous work [9], the cost of each wire is determined by the number of vertices it occupies. In our distance-based objective function, we scale the cost of each wire by summing

up the distances between each pair of consecutive occupied vertices, making the solver aware of the shortest path.

Overall, our *SMTCell* framework is formulated around a relative layered grid graph to efficiently generate vertices and edges on each layer and incorporate distance-based objective formulations and design rule constraints.

3 Cell Pre-Partitioning

Cell partitioning is used in [8] [9] for special cells to reduce the search space for transistor device placement. For example, in D flip-flops, the Clock (Clk), Data-in (Din), Data-out (Dout), and Leader/Follower latches must be arranged in a specific order, such as Din-Leader-Follower-Dout or Dout-Follower-Leader-Din, to optimize the setup time and delay of the flip-flop. Cell partitioning groups transistor devices and encodes their relative positions, thus significantly reducing the search space by limiting the possible placement options.

Table 1. Complexity for DFFHQN_X1 on 2 Fins and 4 Tracks. Denser tracks and vertices increase the problem complexity. Consequently, solving time is increased.

Setting	#variables	#literals	#clauses	runtime (s)
1:1 GR	88846	2302336	242738	377.57
9:5 GR	**116422**	**2472991**	**341133**	**4820.97**

Cell partitioning based on datapath (DP) is defined only for D Flip-flop and is manually coded by designers [8] [9]. By using similar concepts to heuristically partition devices, we can significantly reduce the runtime when dealing with larger cells in general.

Our heuristic comes from the following three observations. First, cell layout solutions under different gear ratio and design rule parameter settings do not shift the order of devices in most cases. Hence, instead of partitioning by function groups, we can encode the exact relative position from a previous solution (each group contains only one device). Second, as we incorporate denser routing tracks, more vertices are created based on Algorithm 2. The number of created vertices is proportional to the number of variables in the SMT formulation [1], as shown in Table 1. Hence, solving such SMT problems requires longer runtimes.

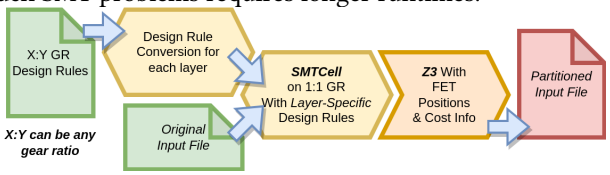

Figure 6. Pre-partitioning approximates the design rule in a 1:1 GR setting and solves for a simpler solution. Then, we extract the necessary information and encode it into a new partitioned input file. This new input guides *Z3* to reduce the solver runtime.

Finally, the cell width in terms of CPP under different cell settings does not drastically increase or decrease, as shown in Table 2 with different cell configurations. For instance, as discussed in Section 2.1, $\max(w_p, w_n)$ gives a lower bound for the cell width. Then, an upper bound on the cell width can be hinted to the solver by using a previous solution with some relaxation by a few (e.g., 2) CPP.

Motivated by these observations, we introduce a two-stage approach in *SMTCell* called Pre-Partitioning:

- First, we solve a relatively simpler problem by utilizing a uniform grid in 1:1 GR with reduced complexity.
- Second, we use the device relative position and cell width value obtained from the first stage to reduce the search space for any given GR. With the relative placement order, the SMT solver only optimizes for routing, which saves runtime.

As our "hint" originates from a solution based on a uniform grid graph, the quality of the initial solution directly influences the quality of the downstream solutions. For instance, given that each layer has its own pitch, design rule constraints prohibit different numbers of vertices on each layer. We are not able to capture such precision in a uniform grid graph. We can enhance the uniform grid graph by assigning different design rule parameters for each layer, rather than applying them universally. These design rule parameters are converted based on the pitch value of each layer from the more difficult GR input. These modifications allow for more accurate design rule considerations and yield a more informative initial solution.

Figure 6 depicts the complete pre-partitioning flow. The partitioned input file, highlighted in red, is passed on to *SMTCell* with GR enablement to obtain a comprehensive cell layout solution.

4 Experiments

Our experiments seek to assess the impact of gear ratio at both cell-level and block-level. We hypothesize that having more *M1* routing resources benefits block-level routing, while vertical alignment between *Poly* and *M1* tracks is preferred at cell-level.

We first investigate the impact of different GR settings, using the 45:30 (3:2) GR as a baseline with various cell designs and track settings. We choose 22 cells with higher drive strength out of the 40 cells available in *SMTCell*. Secondly, we focus on the execution time of layout synthesis for larger cells, where we evaluate the speedup achieved through pre-partitioning. Finally, we present block-level P&R studies to assess the effect of various GR configurations.

Our cell netlists are extracted from the ASAP7 PDK [16] [20]. *SMTCell* flow is run using a single AMD Ryzen 9 5900X Desktop CPU (12-core, 24-thread). We run *Z3* Ver. 4.8.5 with parallel mode enabled [12]. We also enable *SMTCell* to be compatible with the custom process design kit generation flow in PROBE3.0 [5]. Block-level Placement and Routing (P&R) is achieved by using *Cadence Innovus* Ver. 21.1 [21]. *SMTCell* is available in different fin (F) and routing Track

Table 2. Cell metrics and *Z3* runtimes for 45:30/45:27/45:45 GR in 3F5T setting.

Cell Name	#FET	#Net	Cell Width (CPP)			Metal Length (nm)			#Via			#Top Track			Z3 Runtime (s)		
			45:30	45:27	45:45	45:30	45:27	45:45	45:30	45:27	45:45	45:30	45:27	45:45	45:30	45:27	45:45
3 Fins 5 Routing Tracks																	
AND2_X2	6	7	6	6	6	1557	1473	1509	14	15	14	0	0	0	5.66	4.42	6.66
AND3_X2	8	9	7	7	7	1965	1863	2013	20	18	17	0	0	0	10.17	15.77	5.11
AOI22_X1	8	7	6	6	6	1746	1428	1644	15	15	16	0	0	0	26.35	53.84	11.32
AOI22_X2	8	7	11	12	11	4425	4989	4110	39	52	35	0	1	0	265.60	595.37	104.50
BUF_X4	4	5	8	8	8	1980	2010	1857	20	20	17	0	0	0	46.58	33.14	10.21
BUF_X8	4	5	14	14	14	4116	3963	3963	32	31	29	0	0	0	29.56	21.89	5.66
DFFHQN_X1	24	17	19	19	19	7764	7965	8445	61	78	67	1	1	3	803.85	771.76	400.42
INV_X4	2	4	6	6	6	1302	1320	1278	11	11	11	0	0	0	3.615	1.91	1.22
INV_X8	2	4	10	10	10	2766	2790	2742	19	19	19	0	0	0	9.43	9.59	7.45
LHQ_X1	16	13	12	12	12	4617	4599	4869	38	42	41	0	0	2	5602.04	7881.07	2051.69
MUX2_X1	12	12	9	9	9	3375	3396	3195	26	33	25	0	0	2	61.12	58.43	42.01
NAND2_X2	4	6	6	8	6	1710	2496	1686	15	18	15	0	0	0	13.72	32.75	14.30
NAND3_X2	6	8	9	9	9	2766	2781	2694	23	27	23	0	0	0	37.73	31.74	29.79
NAND4_X1	8	10	6	6	6	1524	1275	1671	13	13	13	0	0	0	12.14	9.12	11.41
NAND4_X2	8	10	11	12	11	3546	4464	3978	29	43	33	0	1	0	130.27	286.95	50.24
NOR3_X2	6	8	9	10	9	2766	3330	2694	23	34	23	0	0	0	20.10	17.80	14.52
NOR4_X2	8	10	12	13	12	4635	4731	3774	37	47	33	0	0	0	186.56	243.62	132.61
OAI22_X1	8	10	6	6	7	1701	1758	1755	15	22	15	0	0	0	25.05	9.66	12.46
OAI22_X2	8	10	11	12	11	3753	4803	3774	32	45	33	0	0	0	253.74	225.57	185.25
OR2_X2	8	7	6	6	6	1467	1560	1464	14	15	14	0	0	0	6.01	4.18	5.42
OR3_X2	8	9	7	7	7	1806	1854	1944	17	18	17	0	0	0	12.93	10.50	7.10
XOR2_X1	10	9	7	7	7	2199	2283	2169	18	22	17	0	0	0	18.51	36.35	26.84
Average	-	-	9.00	9.32	9.05	2885.73	3051.41	2874.00	24.14	29.00	23.95	0.05	0.14	0.32	344.58	470.70	142.55

(RT) options: 2F4RT, 3F5RT, and 3F6RT. For brevity, we only show 22 cells in 3F5RT in Table 2.

4.1 Effect of Different Gear Ratio Settings

We present a sensitivity study on gear ratios in Table 2. In this study, we examine the effects of different GR settings while scaling design rules based on the varying pitch values. For instance, by shrinking the pitch value of *M1*, we also shrink design rule parameters on *M2* and *M0* as a reasonable design choice.[13] We use 45:30 (3:2) GR as a baseline. Additionally, we examine 45:27 (5:3) GR (less vertical alignment, more routing resources) and 45:45 (1:1) GR (more vertical alignment, less routing resources).

We assume that vertically aligning *M1* and *Poly* layers is always preferred. This is because our formulation allows vias to be stacked when direct connections within the same net are possible across more than two layers. If the given gear ratio does not allow such a vertical alignment, all connections between *Poly* and *M1* need to be rerouted through *M0*, with two separate via connections and a longer *M0* metal segment in between. This scenario is less desirable, as the *M0* segment is extended due to MAR and VR, potentially exhausting the limited routing resources.

Given that the design rule parameters are scaled with the pitch values, cell-level metrics do not follow an obvious pattern with different GR. Overall, 45:27 (5:3) GR increases cell width, wirelength, and via count. This effect implies some extra detours take place on *M0* as described previously. In particular, cells with larger drive strength (AOI22_X2, NAND4_X2, NOR4_X2 and OAI22_X2) require more CPP in the layout solution.

On the other hand, with 45:45 (1:1) GR, the three largest cells in our library (DFFHQN_X1, LHQ_X1, and MUX2_X1) require extra routing resources from *M2*, leading to blockages in block-level routing. We also observe that more blockages are created when tracks are reduced (i.e., 2F4RT). The limited routing resource on *M1* "overflows" the track usage to *M2*, causing these extra blockages. One notable trend is that the *Z3* runtime and the tightness of *M1* pitch display an inverse relation, implying the necessity of an acceleration scheme for faster design turnaround time.

4.2 Solution Quality with Pre-Partitioning

To evaluate the performance improvement from pre-partitioning on larger cells, we use three cells with the longest runtimes from our experiments: D Flip-Flop, AOI22_X2 and LHQ_X1. We choose 45:25 (9:5) GR as it produces even denser tracks than 45:27 (5:3) GR. By default, D Flip-Flop uses DP partitioning information,[14] while AOI22_X2 and LHQ_X1 do not have any partitioning information. To evaluate the performance impact of pre-partitioning, we perform the following tasks:

- First, we generate these three cells without using pre-partitioning.
- Second, we generate these three cells again, this time with pre-partitioning. The additional runtime of the pre-partitioning flow is added to the total runtime.

[13] Cell designs cannot take advantage of dense routing tracks if design rules are much larger than pitch values.

[14] Clk, Din, Dout, Leader/Follower latches follow the order of Din-Leader-Follower-Dout.

- Finally, if the total runtime exceeds that of the non-partitioned cell layout solution, a "timeout" is recorded.

From Table 3, it is evident that our acceleration approach significantly improves the runtime in most cases.[15] Furthermore, solutions with pre-partitioning have similar quality to those obtained without pre-partitioning. Notably, using the new acceleration technique in *SMTCell*, *Z3* is able to converge to a reasonable solution up to 19 times faster on LHQ_X1 under 45:25 (9:5) GR.

To assess whether pre-partitioning provides a meaningful hint, we perform the following steps to compare the relative positions of transistors and the total cell width:

- First, we solve for 1:1 GR and 9:5 GR standard-cell layouts of larger cells, both without pre-partitioning information.
- Second, we extract the relative positions of PMOS and NMOS, which are encoded into two separate strings.
- Third, we compute the edit distance [10] between 1:1 GR and 9:5 GR to check if they are similar enough for the former to provide informative hints.
- Finally, we also examine the differences in cell width between these solutions to see if the 9:5 GR solution is within the provided guiding range for CPP.

Here, the edit distance is the minimum number of single-character edits required to transform one string into the other. An edit distance of 0 indicates a perfect hint as it implies a matching relative position of transistors between the two GR solutions.

Table 4 demonstrates that between 1:1 GR and 9:5 GR cell layouts, the relative placement orders differ by at most one pair-swap (edit distance = 2). Additionally, the cell width differences remain within the specified range. This further validates the effectiveness of our pre-partitioning flow.

4.3 Block-level Effect

Since the cell-level improvement does not directly contribute to block-level designs due to limited inter-cell routing resources and other factors, we conduct further investigation using our custom cell library and the JPEG Encoder block-level design. We use the same 40 cells described above.

A standard-cell layout contains **local *Poly* and *M1* grids**. A standard-cell row contains **global *Poly* and *M1* grids**. For any standard cell to be legally placed on a standard-cell row, the following requirements must be met:

- Its local *Poly* and *M1* grids must align with the global *Poly* and *M1* grids respectively. (Every red diamond must align with a red dashed line and every blue diamond must align with a blue dashed line in Figure 7.)
- Its left-end and right-end boundaries must align with the global *Poly* grid. (The left-most and the right-most red diamonds must each align with a red dashed line in Figure 7.)

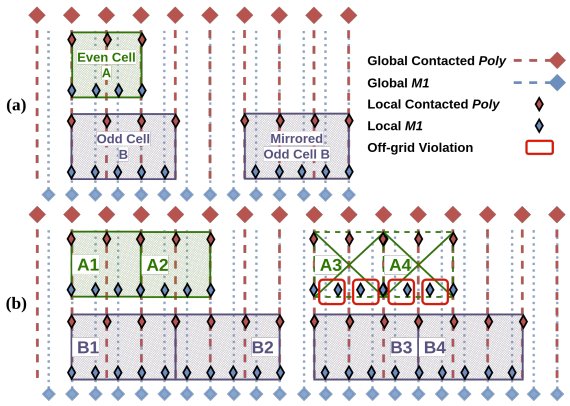

Figure 7. Even and odd cell patterns based on the number of *Poly* they occupy under 3:2 GR. (a) Local *Poly* and *M1* grids (shown in diamonds) must align with the global *Poly* and *M1* grids (shown in dashed lines) to be legally placed. Even cells create one symmetrical pattern. Odd cells create two patterns by mirroring the layout due to their asymmetrical local *M1* grids. (b) Cell legalization% is computed by the placement opportunity for each pattern. *A3* and *A4* cause off-grid violations due to misalignments on *M1* grids. *B2* and *B3* use a mirrored layout to align *M1* grids.

When using a 1:1 GR, all cells can be placed on any column since alignment is guaranteed with a uniform grid. However, with a 3:2 GR, misalignments can occur between the local and global *M1* grids; such misalignments induce off-grid violations.[16]

Figure 7 demonstrates an example under 3:2 GR setting. We classify a cell as having an *even cell (pattern)* or an *odd cell (pattern)* based on the number of *Poly* that it occupies (Figure 7(a)). An odd cell is asymmetrical in terms of the local *M1* grid. By mirroring the layout, an odd cell has two kinds of alignment patterns. Figure 7(b) illustrates that an odd cell can be placed on any column, making it legally placeable at all columns. An even cell has a symmetrical pattern in terms of the local *M1* grid. We cannot create additional alignment patterns by mirroring the layout. Figure 7(b) illustrates that *A3* and *A4* cannot be legally placed due to misalignments on *M1* grids. These misalignments induce off-grid violations. Therefore, an even cell can only be placed at every other column, making it legally placeable at only 50% of the columns (Figure 7(b)). We refer to this calculation as *cell legalization%* (cell legal%).

We further demonstrate that cell legal% directly impacts PPA in block-level design. In Figure 8, we demonstrate two different scenarios when placing different cells adjacent to an odd cell. Due to the legalization issue posed by even cells, *A*1 must be shifted rightward by 1 CPP in Figure 8(a).

[15] LHQ_X1 under 45:30 GR failed to converge within the runtime of its 45:45 GR counterpart. We recognize this as an anomaly that deserves further investigation.

[16] In this context, an off-grid violation arises when a local *M1* grid does not align with any global *M1* grid.

Table 3. Cell metrics and *Z3* runtimes for 45:30/45:25 GR in 2F4T and 3F5T settings. (Exact = MOSFET positions are encoded to be exactly the same as the "hint". DP = Datapath-aware cell partitioning. **Bold** = the better result between no partitioning and pre-partitioning.)

Cell Name	Partition	Cell Width (CPP)		Metal Length (nm)		#Via		#Top Track		Z3 Runtime (s)	
		45:30	45:25	45:30	45:25	45:30	45:25	45:30	45:25	45:30	45:25
2 Fins 4 Routing Tracks											
AOI22_X2	None	12	12	**4176**	**4008**	38	39	2	2	553.49	138.21
AOI22_X2	Exact	12	12	4248	4218	38	**38**	2	2	**22.92 (+33.65)**	**28.30 (+35.67)**
DFFHQN_X1	DP	20	19	7383	7519	**62**	70	**1**	3	2766.67	4820.97
DFFHQN_X1	Exact	**19**	19	**7128**	**7231**	70	70	3	3	**113.55 (+485.46)**	**120.5 (+377.57)**
LHQ_X1	None	12	12	4335	**3867**	38	38	1	1	**6001.01**	8838.66
LHQ_X1	Exact	-	12	-	3908	-	38	-	1	timeout	**37.47 (+419.87)**
3 Fins 5 Routing Tracks											
AOI22_X2	None	11	11	4425	**3951**	39	35	0	0	265.60	595.37
AOI22_X2	Exact	11	11	**4026**	4086	35	35	0	0	**28.70 (+131.41)**	**128.86 (+73.97)**
DFFHQN_X1	DP	19	19	7764	7918	**61**	66	1	0	803.85	771.76
DFFHQN_X1	Exact	19	19	**7731**	**7578**	67	**63**	1	1	**145.01 (+221.19)**	**338.69 (+242.93)**
LHQ_X1	None	12	12	4617	4525	**38**	**37**	0	0	5602.04	7881.07
LHQ_X1	Exact	12	12	**4386**	**4334**	41	40	0	0	**19.16 (+1105.25)**	**34.22 (+1105.25)**

Table 4. Edit distance of PMOS/NMOS relative positions and cell width difference between 45:25 (9:5) GR solution and 45:45 (1:1) GR solution. Ideally, the two results should be as close as possible since 1:1 GR is used to inform the solution space for 9:5 GR. An edit distance of 2 indicates that only one pair of devices is different, while a edit distance of 0 indicates a matching placement order. The cell width difference should be within the guiding range of 2 CPP.

F/RT	Cell Name	PMOS Edit Distance/#PMOS	NMOS Edit Distance/#NMOS	Cell Width Difference (CPP)
2F4RT	AOI22_X2	2/4	**0/4**	**0**
	DFFHQN_X1	**0/12**	2/12	2
3F5RT	AOI22_X2	**0/4**	**0/4**	**0**
	DFFHQN_X1	2/12	**0/12**	2

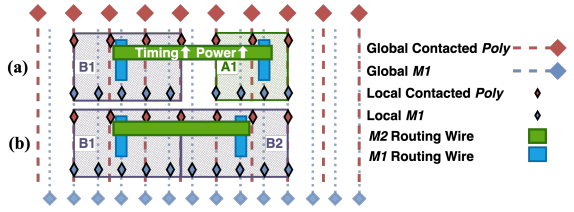

Figure 8. Poor cell legalization% can cause longer routing wires, which increase the timing and power consumption. (a) *A*1 cannot abut *B*1 as this causes an off-grid violation. Shifting *A*1 by 1 CPP can potentially lead to a longer routing wire on *M2*. (b) *B*2 can be placed abutting *B*1, and has a shorter routing wire on *M2*.

*B*2 can be mirrored and placed next to (i.e., abutting) *B*1 in Figure 8(b). The extra CPP caused by legalization stretches the routing wire on *M2* and increases the timing and power consumption.

Table 5 presents a commercial tool report for cell legal% using our cells in block-level design. For each GR, we sort cells from the highest to the lowest usage count. For 1:1 GR setting, all cells are legal and can be placed anywhere, representing a straightforward scenario. However, as GR increases, such as under the 3:2 and 5:3 GR settings, cell legal% becomes increasingly challenging. For the 3:2 GR setting, two kinds of cell patterns can be generated, and half of the most frequently used cells have only 50% cell legalization%. For the 5:3 GR setting, the challenge intensifies, as three kinds of cell patterns may be generated. Indeed, under the 5:3 GR setting, on average only 46% of the columns can be used to legally place cells. With this in mind:

- Since cell legal% directly affects cell placeability and hence routability, we hypothesize that lower cell legal% will degrade timing, power, and area.
- Further, the 1:1 GR setting should yield the best performance, and the 3:2 GR setting should outperform the 5:3 GR setting despite the latter having more routing resources on *M1*.

Figure 9 presents our PPA results comparing the different GR configurations with the JPEG Encoder block-level design. Lower data points indicate better PPA, with the 1:1 GR setting achieving the best result at all clock periods, followed by the 3:2 GR setting, while the 5:3 GR setting ranks as the worst overall. These data align with our our hypothesized impacts of cell legal%.

The differences in PPA metrics become particularly prominent when the effective clock period is as tight as $0.2ns$. The corresponding area values are $2372um^2$, $2461um^2$, and $2623um^2$ for 1:1, 3:2, and 5:3 GR settings respectively (Figure 9(a)). Table 6 shows that with a fixed clock period of $300ps$, increased GR worsens the overall capacitance and wirelength. In conjunction with the increased total cell counts shown in Table 5, this results in higher power consumption for the block. Total power consumption values are $58970nW$, $62720nW$, and $64900nW$ for 1:1, 3:2 and 5:3 GR settings respectively (Figure 9(b)).

Table 5. Comparison of cell legal% in block-level design for different GR. Cell usage is presented with its proportion of the total cell count. Cell legal% is obtained from *Cadence Innovus* (higher % indicates more placement opportunities).

1:1			3:2			5:3		
Cell Name	Cell Usage (Proportion)	Cell Legal%	Cell Name	Cell Usage (Proportion)	Cell Legal%	Cell Name	Cell Usage (Proportion)	Cell Legal%
XOR2_X1	8054 (18%)	100%	INV_X1	6684 (14%)	50%	XOR2_X1	7005 (15%)	33%
AOI22_X1	6507 (14%)	100%	AOI22_X1	6436 (14%)	100%	AOI22_X1	6619 (14%)	67%
NAND2_X1	4644 (10%)	100%	XOR2_X1	5855 (12%)	50%	INV_X1	6437 (14%)	67%
DFFHQN_X1	4420 (10%)	100%	NAND2_X1	4887 (10%)	100%	NAND2_X1	5075 (11%)	33%
INV_X1	4351 (10%)	100%	NOR2_X1	4563 (10%)	100%	DFFHQN_X1	4420 (9%)	33%
NOR2_X1	4070 (9%)	100%	DFFHQN_X1	4420 (9%)	50%	NOR2_X1	3790 (8%)	33%
Total #Cell (Avg. Legal%)	45469	100%	Total #Cell (Avg. Legal%)	47110	74%	Total #Cell (Avg. Legal%)	47060	46%

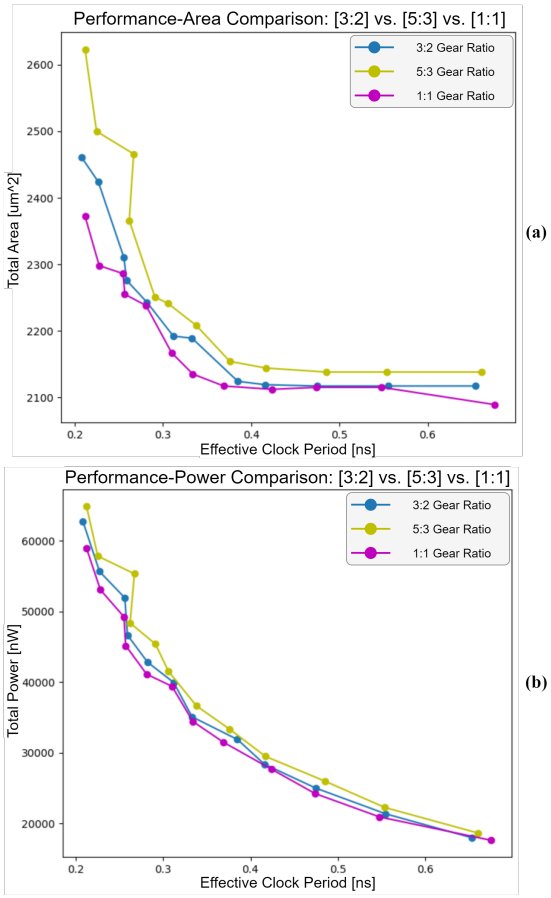

Figure 9. Block-level PPA results under different gear ratios. (a) 5:3 GR has the largest total area, followed by 3:2 GR. 1:1 GR has the smallest total area. (b) 5:3 GR consumes the most total power, followed by 3:2 GR. 1:1 GR consumes the least total power.

In the above PPA results for the JPEG Encoder designs (Figure 9), the 1:1 GR setting outperforms others, emphasizing the crucial role of cell legal% in influencing overall performance metrics, including area and power consumption. Consequently, to effectively demonstrate the advantages of a finer GR, enhancing cell legal% becomes imperative.

Table 6. As gear ratio increases, capacitance increases due to the denser tracks and longer wires. These factors impact the overall power, performance, and area.

Gear Ratio	1:1 GR	3:2 GR	5:3 GR
Total Cap (pF)	39.51	42.42	44.61
Total Wirelength (um)	62992	72154	79159

These findings motivate us to continue investigating the impact of variant GR values, as we can generate multiple copies of the same functional cells with different offsets, as shown in Figure 10. By offsetting the local *M1* grid from the left edge of the cell boundary for even cell layouts, we create an additional alignment pattern for even cells in Figure 10(a). As shown in Figure 10(b), the additional copy of an even cell layout can be used by *A3* and *A4*, thus increasing the cell legal% by offering more flexibility in placement and routing. This flexibility not only enhances the efficient utilization of the available layout space but also contributes to reduced wirelength and improved routing efficiency. Consequently, PPA metrics benefit from reduced power consumption, improved signal propagation, and more compact layouts – i.e., better overall block-level design outcomes.

5 Conclusion

We have tackled the challenge in standard-cell library creation with gear ratio by leveraging and enhancing Satisfiability Modulo Theory (SMT) formulations. Our approach enables automation of the standard-cell layout synthesis process, accommodating varying gear ratio settings. Our framework handles the fine-grained layout design constraints and interactions that emerge in difficult gear ratio settings, which present substantial challenges for human designers. We have also conducted comprehensive studies at both cell-level and block-level. These studies identify the best-performing cells and analyze footprint shrinkage, and moreover confirm that a mere reduction in *M1* pitch does not lead to proportional improvements in block-level scaling. This realization underscores the need for careful tuning of design rule parameters to fully unlock their potential for optimized cell designs.

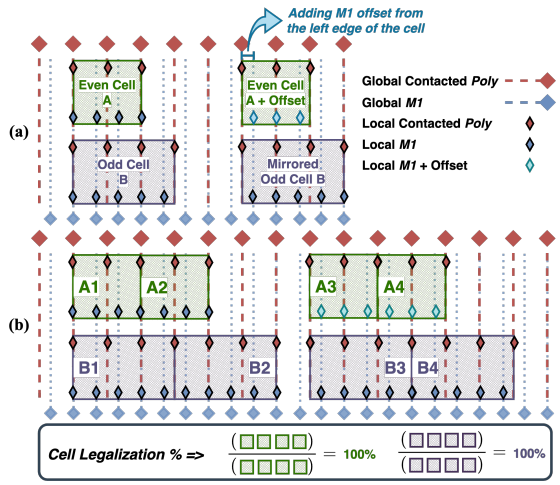

Figure 10. Extending from Figure 7, we introduce an offset from the left edge of the local *M1* grid for even cells. (a) For any even cell layouts under the same function, we solve for two different copies of the cell: one without *M1* offset (shown in dark blue diamonds) and one with *M1* offset (shown in light blue diamonds). (b) *A*3 and *A*4 take advantage of the local *M1* offset copy to align with the global *M1* grid. By introducing the offset, even cells become legally placeable at every column.

Our ongoing research focuses on determining the ideal vertical routing pitch by empirical studies, rather than relying solely on exhaustive searches. As shown in our experiments, GR introduces a trade-off between more routing resources and potential timing and power burdens. Future cost-benefit studies with more fully-elaborated cell libraries and performance models can uncover the underlying impacts of GR tuning. IR-drop comparison according to various PDN schemes is additionally required for holistic block-level evaluation. These additions to *SMTCell* can potentially lead to more effective cell layout generation processes and to enhance the accuracy of existing DTCO flows for pathfinding [4][5].

Acknowledgments

This work is partially supported by Logic Pathfinding Lab, Samsung Semiconductor Inc., NSF grants CCF-2110419 and CCF-2112665, DARPA HR0011-18-2-0032, and the C-DEN center.

References

[1] C. W. Barrett, R. Sebastiani, S. A. Seshia and C. Tinelli, "Satisfiability Modulo Theories", *Handbook of Satisfiability*, IOS Press, 2021. https://api.semanticscholar.org/CorpusID:249494188

[2] C.-K. Cheng, *et al.*, "Complementary-FET (CFET) Standard Cell Synthesis Framework for Design and System Technology Co-Optimization Using SMT", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29(6) (2021), pp. 1178–1191. https://doi.org/10.1109/TVLSI.2021.3065639

[3] C.-K. Cheng, C.-T. Ho, D. Lee and B. Lin, "Monolithic 3D Semiconductor Footprint Scaling Exploration Based on VFET Standard Cell Layout Methodology, Design Flow, and EDA Platform", *IEEE Access* 10 (2022), pp. 65971–65981. https://doi.org/10.1109/ACCESS.2022.3184008

[4] C. Chidambaram, *et al.*, "A Novel Framework for DTCO: Fast and Automatic Routability Assessment with Machine Learning for Sub-3nm Technology Options", *Proc. IEEE Symposium on VLSI Technology*, 2021, pp. 1–2.

[5] S. Choi, *et al.*, "PROBE3.0: A Systematic Framework for Design-Technology Pathfinding with Improved Design Enablement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, to appear (2023). https://doi.org/10.1109/TCAD.2023.3334591

[6] A. B. Kahng, L. Wang and B. Xu, "The Tao of PAO: Anatomy of a Pin Access Oracle for Detailed Routing", *Proc. ACM/ESDA/IEEE Design Automation Conference*, 2020, pp. 1–6. https://doi.org/10.1109/DAC18072.2020.9218532

[7] I. Kang, C. Han, D. Park and C.-K. Cheng, "Fast and Precise Routability Analysis with Conditional Design Rules", *Proc. ACM/IEEE International Workshop on System Level Interconnect Prediction*, 2018, pp. 1–8. https://doi.org/10.1145/3225209.3225210

[8] D. Lee, "Logical Reasoning Techniques for VLSI Applications", *Ph.D. Dissertation*, University of California, San Diego, 2022.

[9] D. Lee, *et al.*, "SP&R: SMT-Based Simultaneous Place-and-Route for Standard Cell Synthesis of Advanced Nodes", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40(10) (2021), pp. 2142–2155. https://doi.org/10.1109/TCAD.2020.3037885

[10] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals", *Soviet physics. Doklady* 10 (1965), pp. 707–710.

[11] S. Maheshwaram, *et al.*, "Vertical Silicon Nanowire Gate-All-Around Field Effect Transistor Based Nanoscale CMOS", *IEEE Electron Device Letters* 32 (2011), pp. 1011–1013. https://doi.org/10.1109/LED.2011.2157076

[12] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver", *Proc. International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340. https://doi.org/10.1007/978-3-540-78800-3_24

[13] G. Northrop. "Design Technology Co-optimization in Technology Definition for 22nm and Beyond", *Proc. Symposium on VLSI Technology*, 2011, pp. 112–113.

[14] D. Park, *et al.*, "ROAD: Routability Analysis and Diagnosis Framework Based on SAT Techniques", *Proc. International Symposium on Physical Design*, 2019, pp. 65–72. https://doi.org/10.1145/3299902.3309752

[15] E. Park and T. Song, "Complementary FET (CFET) Standard Cell Design for Low Parasitics and Its Impact on VLSI Prediction at 3-nm Process", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 31 (2023), pp. 177–187. https://doi.org/10.1109/TVLSI.2022.3220339

[16] V. Vashishtha, M. Vangala and L. T. Clark, "ASAP7 Predictive Design Kit Development and Cell Design Technology Co-optimization: Invited paper", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2017, pp. 992–998. https://doi.org/10.1109/ICCAD.2017.8203889

[17] X. Xu, *et al.*, "Self-Aligned Double Patterning Aware Pin Access and Standard Cell Layout Co-Optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34 (2015), pp. 699–712. https://doi.org/10.1109/TCAD.2015.2399439

[18] L. Liebmann, *et al.*, "Exploiting regularity: breakthroughs in sub-7nm place-and-route", *Proc. Design for Manufacturability through Design-Process Integration XI* 10148, SPIE, 2017

[19] SMTCellUCSD: Cell Layout Generation for DTCO/STCO Exploration Toolkit. https://github.com/ckchengucsd/SMTCellUCSD

[20] ASAP7 PDK and Cell Libraries. https://github.com/The-OpenROAD-Project/asap7

[21] Cadence Innovus User Guide. https://www.cadence.com