Au-MEDAL: Adaptable Grid Router with Metal Edge Detection And Layer Integration

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

Seokhyeong Kang
POSTECH
shkang@postech.ac.kr

Sehyeon Kim
POSTECH
sehyeon2001@postech.ac.kr

Jakang Lee
UC San Diego
jal216@ucsd.edu

Dooseok Yoon
UC San Diego
d3yoon@ucsd.edu

***Abstract*—Standard-cell layout routing faces several challenges due to stringent design rules. In this paper, we propose Au-MEDAL, a new SMT-based standard-cell router that supports a rich set of advanced-node design rules at the nanometer scale. Au-MEDAL implements extended design rules to enable bidirectional routing and proposes methods that: (i) ensure at least one pin access point for block-level routing; (ii) integrate Middle-of-Line (MOL) and Back-End-of-Line (BEOL) routing; (iii) support variable routing grid spacings; and (iv) perform off-grid design rule checks. As a result, based on the ASAP7 PDK [17], Au-MEDAL achieves cell-level DRC-clean routing unattainable by previous in-cell routers, Au-MEDAL also improves pin accessibility and reduces the number of cells that use M2, thereby increasing feasible access scenarios during P&R. Across the evaluated block designs, Au-MEDAL achieves improvements of 0.87%, 1.19%, 2.34%, and 25.05% in total core area, total power, effective clock, and total negative slack, respectively. Its block-level DRC performance is comparable to that of manually designed cell layouts and shows significant improvements over previous in-cell router results.**

I. Introduction

Recently, demand for standard-cell layout in advanced nodes has increased. However, generation of standard-cell layouts is time-consuming and labor-intensive. To overcome this challenge, various works have explored automated standard-cell layout generation. Research on automated standard-cell layout can be broadly divided into two main approaches: (1) *simultaneous methods*, where transistor placement and standard-cell layout routing, also known as in-cell routing, are performed simultaneously; and (2) *sequential methods* that perform transistor placement first, followed by in-cell routing. [1]–[5] use a Satisfiability Modulo Theories (SMT) solver to perform simultaneous transistor placement and in-cell routing. The consideration of placement and routing (P&R) constraints at the same time improves layout quality. However, computation time grows exponentially with cell size, limiting scalability. Thus, other works [6]–[14] employ sequential methods to improve scalability.

Despite many previous works, there is room for improvement in the automatic generation of standard-cell layouts. (1) Prior methods have limitations in applying bidirectional routing methods that fully comply with design rules. While previous works [9], [12], [15] have adopted bidirectional routing, they do not differentiate between side and tip types of metal edges. Hence, these methods violate minimum spacing rules and cannot ensure DRC-clean layouts. (2) The works [9], [12], [15] also do not take into account DRC violations (DRVs) that may occur between different layers. Consequently, their approach—where MOL routing is performed first, followed by BEOL routing—limits the router's ability to explore and optimize within the feasible routing space. (3) Prior works typically rely on routing strategies based on uniform or fixed grid spacing, which restricts routing grid flexibility. Fixed routing grids make it difficult to generate layouts that meet user-defined specifications and also hinder the exploration of more optimized solutions.[1] (4) Last, prior works often fail to ensure pin accessibility. Even if the cell is internally DRC-clean, one or more DRC violations may still occur during block-level routing due to blocked pins. [2], [16] consider individual pin accessibility at the cell design stage, but do not guarantee routability at the block-level.

This paper proposes **Au-MEDAL**, a new SMT-based standard-cell layout router that overcomes the above limitations. Our contributions:

[1]ASAP7 [17] assumes EUV single patterning for the M1–M3 routing layers, allowing for bidirectional routing and flexible routing track assignment.

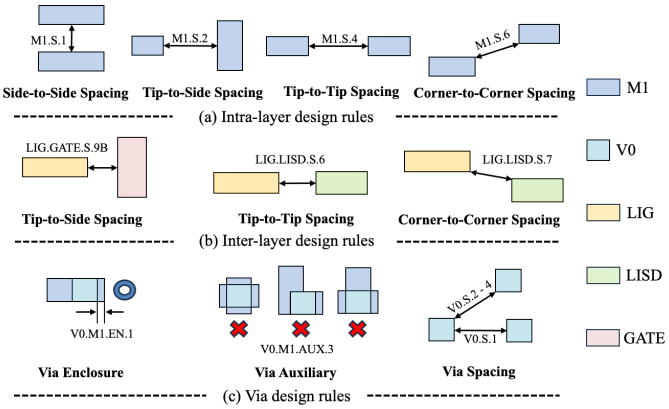


Fig. 1: (a) shows the intra-layer design rules, illustrating various spacing rules of M1 layer; (b) presents the inter-layer design rules, demonstrating the design rules between different layers; and (c) displays the design rules related to vias.

- We incorporate metal edge detection to perform bidirectional routing that requires extended design rules, achieving cell-level DRC-clean for all generated cells.
- We route MOL and BEOL layers simultaneously and adjust the routing grid per layer, reducing the number of cells using M2 metal by 61% on average compared to previous methods.
- We accelerate large cell generation by removing unnecessary routing grids, and in the BUFx24 experiment, we reduce runtime by 77% while maintaining layout quality.
- We maximize pin stretchability and guarantee at least one valid routing point for each pin, reducing block-level DRVs by an average of 92% compared to previous in-cell routers.
- Our code and data are available at [22].

II. Background and Preliminaries

A. Design Rules for Standard Cells

Design rules for standard-cell layout fall into three categories: intra-layer design rules defined within the same metal layer; inter-layer design rules defined between different layers; and via design rules.

1) Intra-layer Design Rules: Many design rules are defined for elements on the same layer. Fig. 1(a) shows the most common rule, the minimum spacing design rule, which is specified in nanometer (nm) scale. ASAP7 [17], the PDK used in this work, supports bidirectional routing, hence design rules are applied differently depending on the type of metal edge (i.e., side, tip, or corner). Minimum metal area rules are also defined. Because all design rules are specified in nm units, it is crucial that the router can accurately determine the type of metal edge and measure the distance between metals precisely at the nm scale.

2) Inter-layer Design Rules: Some design rules are defined across different layers. Fig. 1(b) illustrates representative inter-layer design rules defined in ASAP7. Local Interconnect Gate (LIG) and Local Interconnect Source Drain (LISD) layers serve as MOL routing layers and have their own intra-layer design rules (such as minimum metal width and minimum spacing). However, these two layers are not independent because they are at the same physical height. This

TABLE I: Notations for SMT formulation.

Term	Description
$G(V, E)$	Three-dimensional routing grid
L	Set of metal layers used for routing
V	Set of vertices in the routing grid G
V_l	Set of vertices in metal layer l of the routing grid G ($l \in L$)
v	Vertex at the coordinate (x_v, y_v, z_v), where z_v is the metal layer index
H	Set of possible directions in 2D space: {left, right, front, back}
$\hat{d}$	Direction vector
a	Axis, which can take one of two values: {horizontal, vertical}
$m_{v,u}$	Metal edge connecting two vertices v and u
$m_v^{\hat{d}}$	$\hat{d}$-directional metal edge from vertex v
cm_v^a	A variable indicating metal crossing the a-axis at vertex v
$span(P)$	Length of the consecutive linear segment pattern P from n_1 to n_2
$t_{v,\hat{d}}$	$\hat{d}$-directional tip variable that exists in vertex v
$s_{v,\hat{d}}$	$\hat{d}$-directional side variable that exists in vertex v
$c_{v,\hat{d},\hat{d}^\perp}$	$(\hat{d}, \hat{d}^\perp)$-directional (order-independent) corner variable that exists in vertex v
$via_{v,\hat{d}}$	$\hat{d}$-directional via variable that exists in vertex v
$enc_{v,\hat{d},a}$	a-axis enclosure metal variable caused by $\hat{d}$-directional via at vertex v

introduces a risk of shorts and necessitates inter-layer design rules to meet specific process requirements.

3) Via Design Rules: In addition to the minimum spacing rule, various design rules are defined for vias. Fig. 1(c) illustrates several via rules. The via enclosure rule specifies the metal extension required due to the via, while the via auxiliary rule defines the conditions for configuring the via relative to adjacent metal.

B. SMT Formulations of Design Rules

Since we use an SMT solver for in-cell routing, we convert various design rules into SMT constraints. These constraints are expressed on a grid, and can be classified into two categories: network flow constraints and design rule constraints. We now explain design rule constraints using the minimum tip-to-tip (T2T) and corner-to-corner (C2C) spacing rules as examples. See [1] for more details on network flow constraints and related topics.

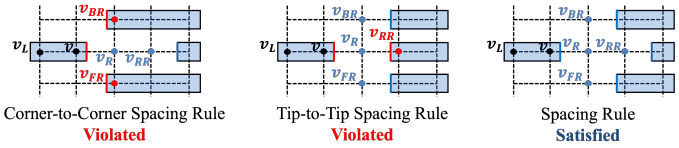


Fig. 2: Minimum spacing rules for horizontal routing layers.

1) On-grid Design Rule Check: To account for various design rules in the routing, geometric variables, as listed in Table I, are introduced. One such variable is $t_{v,\hat{d}}$, which distinguishes the tip of a metal. When the tip geometric variable is *True*, it signifies that the corresponding vertex is a metal tip in the specified direction. For example, the right tip variables of vertex v and metal variables can be expressed as follows: $(t_{v,R} = \neg m_{v,v_R} \wedge m_{v_L,v})$. Once the tip variables are defined for all vertices and directions, the relationships among them can be used to encode design rules as constraints. For example, to enforce the minimum spacing (MS) rule between metal tips, the minimum spacing between tips is expressed in the number of grids, and a constraint is added to ensure that no other metal exists within that range. By using these constraints, an SMT solver can find routing solutions that satisfy the MS design rules.

Fig. 2 shows how the grid-based design rule is applied to the right metal tip when the minimum grid count for the MS rule is set to 2 in the horizontal routing metal layers. To apply the minimum C2C spacing rule in the right direction at v, routing is restricted so that the left metal tip is forbidden at v_{BR} or v_{FR} (see the left side of Fig. 2). Similarly, the minimum T2T spacing rule forbids a left tip at v_R and v_{RR} (see the middle side of Fig. 2).

$$MS_{R_{t2t}}(v) = AM\text{-}1(t_{v,R}, t_{v_R,L}) \wedge AM\text{-}1(t_{v,R}, t_{v_{RR},L}),\ \forall v \in V \quad (1)$$

$$MS_{R_{c2c}}(v) = AM\text{-}1(t_{v,R}, t_{v_{FR},L}) \wedge AM\text{-}1(t_{v,R}, t_{v_{BR},L}),\ \forall v \in V \quad (2)$$

$$MS_R(v) = MS_{R_{t2t}}(v) \wedge MS_{R_{c2c}}(v) \quad (3)$$

$$MS = \bigwedge_{v \in V} (MS_L(v) \wedge MS_R(v)) \quad (4)$$

Eqs. (1-2) illustrate how the variables for MS design rules are assigned to the right tip of the grid. In these equations, *AM*-1 stands for "At most 1," meaning that among the variables in the condition, at most one can be *True*. Although the example focuses only on the right tip, the same approach is extended to the front, back, and left tips. Eqs. (3-4) show how these variables are integrated into the MS rule at each vertex of the horizontal layers. If the layer is vertical, the directions are defined as front and back instead of left and right. Under these constraints, the SMT solver guarantees that the routing solution never violates T2T and C2C spacing rules.

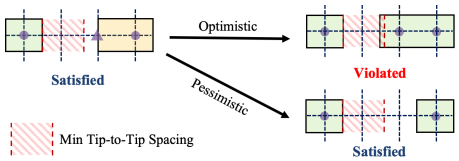


Fig. 3: Converting off-grid objects to on-grid for design rule checks: inefficiencies arise during alignment of off-grid objects to the grid.

2) Off-grid Design Rule Check: On-grid design-rule check has the advantage of easily enforcing rules when objects are placed exactly on the routing grid. However, off-grid objects cannot be checked in this way. As shown in Fig. 3, an off-grid object occupies 1.5 grids. If we align the object strictly to the routing grid, design-rule checking becomes possible, yet introduces two risks: (1) if we optimistically treat the object as occupying only one grid, the layout result may incur DRVs; and (2) if we pessimistically assume that it spans two grids, the SMT solver may not find a feasible solution.

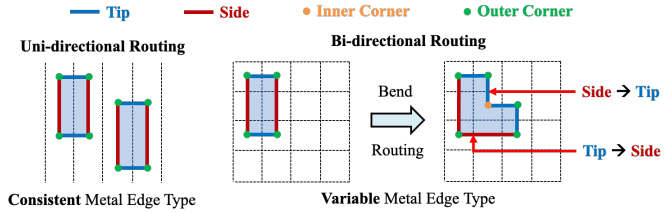
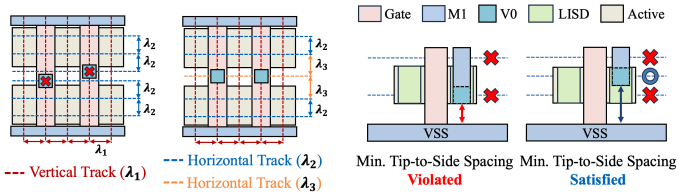


Fig. 4: Unlike unidirectional layers, bidirectional layers allow the tip and side edges to vary depending on the metal pattern. In addition, a new edge type, the inner corner, is introduced.

C. Bidirectional Routing

Bidirectional routing reduces the number of metal layers needed inside a cell while greatly increasing routing freedom. Unlike traditional unidirectional routing, which uses vias to change direction, bidirectional routing can effectively avoid the parasitic effects and reliability issues introduced by vias. However, applying the design rules for bidirectional routing within an in-cell router is not straightforward. Unidirectional routing has a predetermined orientation, making it easy to distinguish between a "tip" edge and a "side" edge. As shown in Fig. 4, on a unidirectional (vertical) layer the left/right edges are always sides and the top/bottom edges are always tips. In contrast, bidirectional routing supports both vertical and horizontal tracks, so there is no fixed routing direction per layer. Fig. 4 illustrates that when a horizontal metal segment is added, the tip and side edges can be interchanged. As shown in Fig. 1, the MS rule to apply is determined by the type of metal edge, so accurate distinction is essential.



(a) Non-uniform routing grid spacing (b) Fine routing grid spacing

Fig. 5: The advantages of flexible routing grid spacing.

D. Flexible Routing Grid Configuration

In-cell routers typically employ a three-dimensional routing grid, and the routing quality can vary depending on how the grid is constructed.

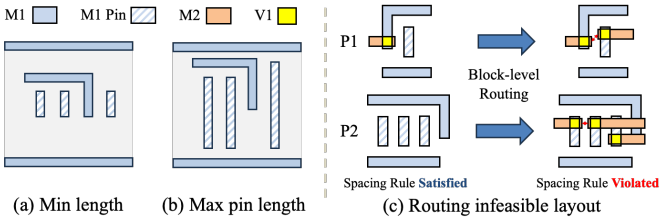


Fig. 6: Metric evaluation of various standard-cell layouts: (a) layout with only wire length minimization; (b) layout with maximized pin metal length; and (c) layout infeasible for block-level routing.

1) Non-uniform Routing Grid Spacing: Generally, routing grids are defined based on the metal pitch of the layer. However, with in-cell routing, there are often cases where using a non-uniform routing grid proves to be more effective. Fig. 5(a) illustrates the use of non-uniform routing tracks. The figure illustrates a case where a horizontal routing track for gate contact (denoted by a orange dotted line) is added at the center. In this case, while the blue dotted lines maintain uniform spacing, the spacing between the orange and blue dotted lines is defined differently. If the in-cell router relies solely on uniform grid spacing in such cases, routing failures may occur due to insufficient access to gate contacts, or overhead—such as the incorporation of extra layers—may be required.

2) Fine Routing Grid Spacing: In in-cell routing, additional routing tracks with a spacing smaller than the minimum metal pitch are often required. These tracks can enable a valid layout in cases where a solution cannot be found using the standard routing grid. This is illustrated in Fig. 5(b), which presents a case where a denser routing grid enables a valid layout. In the figure, routing tracks at the default metal pitch can cause S2T spacing rule violations between the M1 "VSS" and signal metal. By placing routing tracks more finely, a valid solution becomes feasible.

As shown in the two cases above, the spacing of routing grids may vary depending on the conditions, so as to enable better (or valid) solutions. In such cases, it becomes challenging to apply design rule checking based solely on the number of grid units, because the spacing of a single grid unit can vary depending on its location and orientation.

E. Standard-Cell Routing Evaluation Metrics

Standard-cell routing quality is assessed using three key metrics. (1) *Electrical characteristics*, namely, the cell's power consumption and delay, which are both influenced by in-cell routing. (2) *Pin accessibility* measures how easily the block-level router can reach each pin; poor pin accessibility limits viable routing scenarios and often leads to numerous DRVs at the block level. (3) *Routing resource availability* reflects how much of the cell's unused metal-track and via resources can be exploited during block-level routing. if availability is low, nets are forced to detour around congested regions, degrading overall performance.

Fig. 6 demonstrates the importance of pin accessibility across several layouts. Fig. 6(a) and (b) both show layouts for the same cell, with (a) representing a layout that minimizes overall metal length, while (b) maximizes pin metal length. The layout in (a) suffers from insufficient pin access points and limited pin extension distance, making DRC-clean routing challenging with block-level design tools. By contrast, the layout in (b) provides ample pin access points, enabling DRC-clean routing at the block-level.

Additionally, Fig. 6(c) illustrates how a standard-cell layout can be DRC-clean, yet violate DRC after block-level routing. The example P1 uses M2 metal in the in-cell routing, which causes DRV at every access point of the input pin at the block level. The example P2—despite not using the M2 layer at all—has no valid routing path that can connect all pins at the block level. These examples clearly show why standard-cell layouts must be designed with multiple metrics taken into account simultaneously.

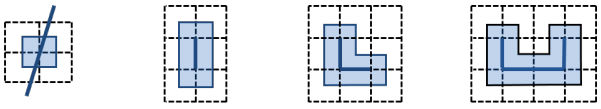


Fig. 7: The four routing patterns used to distinguish between the side and tip of metal edges.

III. Proposed Methods

Our Au-MEDAL is capable of processing design specifications and rules at the nm scale, and implements a variety of features that existing approaches do not support. Tables II and III compare Au-MEDAL with leading previous approaches, with the former comparing supported features and the latter focusing on design rules that can be handled as user-defined input parameters. As shown in Section IV, Au-MEDAL is the first (academic, open-source) in-cell router that achieves DRC-clean layouts, per *Siemens Calibre v23.3*, in the ASAP7 PDK.

A. Extended Design Rules

1) Metal Edge Type Detection: We propose a metal edge type detection approach to accommodate various design rules required for bidirectional routing. This method is implemented by defining new geometric direction variables in each grid, thereby enabling the effective enforcement of different minimum spacing design rules such as S2S, S2T, T2T, and C2C. The variables used in this approach are defined as $t_{v,\hat{d}}$, $s_{v,\hat{d}}$, and $c_{v,\hat{d},\hat{d}^\perp}$ in Table I. These variables depend on the routing pattern.

$$P^v_\bullet = \bigvee_{l \in Layers_{adj(v)}} m_{v,v_l}, \tag{5}$$

$$P^{v,n,\hat{d}}_I = \bigwedge_{i=v-n_1}^{v+n_2-1} m_i^{\hat{d}^\perp} \wedge \bigwedge_{i=v-n_1}^{v+n_2} \neg m_i^{\hat{d}}, \tag{6}$$

$$P^{v,n,\hat{d}}_{L_{upper}} = m^{\hat{d}}_{v+n_2} \wedge \bigwedge_{i=v-n_1}^{v+n_2-1} m_i^{\hat{d}^\perp} \wedge \bigwedge_{i=v-n_1}^{v+n_2-1} \neg m_i^{\hat{d}}, \tag{7}$$

$$P^{v,n,\hat{d}}_U = m^{\hat{d}}_{v-n_1} \wedge m^{\hat{d}}_{v+n_2} \wedge \bigwedge_{i=v-n_1}^{v+n_2-1} m_i^{\hat{d}^\perp} \wedge \bigwedge_{i=v-n_1+1}^{v+n_2-1} \neg m_i^{\hat{d}}, \tag{8}$$

$$\text{where } n_1 + n_2 = n,\ n_1, n_2 \geq 0,\ n, n_1, n_2 \in \mathbb{Z}$$

Eqs. (5-8) illustrate several commonly used routing patterns in mathematical form. P denotes a pattern, and the subscripts $\bullet$, I, L, and U indicate the shape of each pattern. These shapes are also shown in Fig. 7. Additionally, in Eq. (5), v_l represents a vertex with the same x and y coordinates in adjacent layers—for example, if v is on M2, the v_l may be on M1 or M3. While this primarily denotes a via, it also includes cases where no physical via is created, such as the connection between LIG and LISD. P_L can be expressed as $P_{L_{upper}} \vee P_{L_{lower}}$, where $P_{L_{lower}}$ can be inferred from Eq. 7.

$$s_{v,\hat{d}} = \bigvee_{span(P) \geq len} P^{v,n,\hat{d}}_I, \tag{9}$$

$$t_{v,\hat{d}} = \neg\left(m^{\hat{d}}_v \vee m^{-\hat{d}^\perp}_v \vee m^{\hat{d}^\perp}_v\right) \wedge \left(m^{-\hat{d}}_v \vee P^v_\bullet\right) \vee \bigvee_{P \in \{P_I, P_L, P_U\}} \bigvee_{span(P) < len} \left(P^{v,n,\hat{d}} \wedge \neg M_{eop(P)}\right), \tag{10}$$

$$c_{v,\hat{d},\hat{d}^\perp} = \left(m^{\hat{d}}_v \wedge m^{\hat{d}^\perp}_v\right) \vee \left(m^{-\hat{d}}_v \vee m^{-\hat{d}^\perp}_v \vee P^v_\bullet\right), \tag{11}$$

$$\text{where } v \in V,\ \hat{d} \in H$$

Eqs. (9-11) respectively illustrate the use of routing patterns in Eqs. (5-8) and logical operations on metal edges to express (or, detect) side, tip, and corner variables. The end of each pattern is denoted by $M_{eop(P)}$. For the I pattern, the end is defined at both ends of the span, while for the L pattern, it is defined at the end in the $\hat{d}$. No explicit end is defined for the U pattern. Additionally, len is the minimum side length and w is the width of layer l. These variables are used to apply MS rules.

TABLE II: Features supported by open-source in-cell routers. Δ indicates partial support.

Features	Csyn-fp [18]	SMTCell [3]	Au-MEDAL
Distance-based rule checking	X	✓	✓
Metal edge type detection	X	X	✓
Enclosure-aware rule check	X	✓	✓
Bidirectional routing	Δ	X	✓
Flexible grid spacing	X	Δ	✓
Applying min I/O pin length	X	✓	✓
Inter-layer rule-aware routing	X	X	✓
Off-grid design rule check	X	X	✓

TABLE III: Design rules supported by open-source in-cell routers. Δ indicates partial support.

Design Rule Parameter	Csyn-fp [18]	SMTCell [3]	Au-MEDAL
min layer width	X	✓	✓
min S2S spacing	X	✓	✓
min S2T spacing	X	X	✓
min T2T spacing	X	✓	✓
min C2C spacing	X	Δ	✓
min metal area	X	✓	✓
min via T2T spacing	X	✓	✓
min via C2C spacing	X	X	✓
min metal enclosure	X	Δ	✓

2) Extended Minimum Spacing Rule: Using the metal-edge type detection logic described above, Au-MEDAL adds SMT constraints for side, tip, and corner on every grid (see Section II-B). It also handles spacing rules introduced by via enclosures. As an example, we explain how to enforce the minimum C2C constraint when a via enclosure is present. Assume two vertices, u and v, whose C2C spacing already satisfies the minimum C2C rule; even if metal exists at both vertices, the design rule holds, so no additional SMT constraint is needed. However, if either u or v contains a via, a metal enclosure forms at that location, which can violate the spacing rule. In this case, we add the following SMT constraint:

$$MS_{c2c}(u,v) = \neg(enc_{u,up,hor} \vee enc_{v,up,hor}) \vee AM\text{-}1(c_{u,B,R}, c_{v,F,L}),$$

where enc denotes the enclosure created by the via, up means that the corresponding enclosure is generated by a via extending upward from the vertex, and hor indicates that the enclosure is formed in a horizontal direction. This equation limits the formation of the corner variables—specifically, the ones at the closer corners—when a horizontal enclosure is formed on one side. Moreover, by computing via enclosures in all directions across upper and lower layers and incorporating them into constraints, all spacing rules can be extended.

3) Inter-layer Design Rules: To consider design rules between different layers, we extend the range of grids we traverse to cover other layers. If a minimum spacing is defined between these different layers, we add constraints to enforce the spacing rules.

4) Via Design Rules: We use via variables to apply via design rules. The $via_{v,\hat{d}}$ variable is decided by vertex v and direction $\hat{d}$, which denotes that a via exists in direction $\hat{d}$ from vertex v (for example, if $\hat{d}$ is "up" for a vertex in the M1 layer, the via variable represents V1). By using via variables, we can similarly apply the spacing rules mentioned earlier, so here we highlight the constraints used for via design rules, excluding the spacing rules.

$$ENC(v) = \bigwedge_{\hat{d}\in\{up,down\}} \left(\neg via_{v,\hat{d}} \vee enc_{v,\hat{d},hor} \vee enc_{v,\hat{d},ver}\right) \quad (12)$$

$$AUX(v) = \bigwedge_{\substack{\hat{d}\in\{up,down\}\\ a\in\{hor,ver\}}} \left(\neg enc_{v,\hat{d},a} \vee \neg enc_{v,a^{\perp}} \vee \neg cm_{v}^{a^{\perp}}\right) \quad (13)$$

Eqs. (12-13) demonstrate how to specify various design rules that stem from vias. In this context, $enc_{v,a^{\perp}}$ denotes $enc_{v,\hat{d},a^{\perp}} \vee enc_{v,-\hat{d},a^{\perp}}$. The ENC rule indicates that if a via is present, either a horizontal enclosure or a vertical enclosure must be generated. AUX disallows the concurrent presence of vertical and horizontal metals crossing the via. All of these conditions are illustrated in Fig. 1(c). Moreover, the via corresponds to the z-axis metal in the routing graph; cases where different layers (e.g., LIG and LISD) are used without creating an actual physical via are excluded. We extend these conditions to all vertices and incorporate them as design constraints that must be met, similar to MS.

B. Off-grid Design Rule Check

Au-MEDAL applies off-grid design rule checking in the following three scenarios. (1) When a layer used for placement is also used for routing. In advanced nodes, the LISD layer must always reside above the active region, so LISD routing must take into account the LISD determined during the placement. (2) When a placement-only layer is still subject to routing design rules. For example, as shown in Fig. 1(b), the Gate layer itself is not a routing layer, but because there is a MS rule between Gate and the routing layer LIG, LIG routing must also consider the Gate decided during the placement. (3) When treating objects as off-grid yields reduced computational overhead. By excluding the power nets (VDD, VSS) from routing and fixing the power rails during placement—then treating those rails as off-grid objects—the router's overhead can be significantly lowered.

To realize off-grid design rule checking, we perform three steps in the following order. First, objects fixed prior to routing are stored as nm-scale polygons. Second, each routing grid is iterated through to identify any polygons within the grid that may violate the minimum spacing rules. Third, if a minimum spacing rule violation is detected, the associated metal edge variable for that routing grid is forced to *false*. For example, if there is an off-grid object designated as side to the right of vertex v, and this object violates the minimum S2S rule with the object generated at vertex v, then a constraint, $s_{v,R} = false$, is added to the constraints.

C. Flexible Routing Grid Spacing

Au-MEDAL supports flexible routing grid functionality, allowing us to flexibly apply finer or coarser grids relative to the default routing grid spacing. In this section, we discuss when and why this flexibility can be beneficial. The default routing grid spacing is set to M1 Pitch in the vertical direction and half of CPP in the horizontal direction.

On one hand, a finer routing grid can be used to achieve superior layout solutions. For example, Au-MEDAL accepts the Minimum Pin Length (MPL) [1] in nm scale as a parameter for standard cells. Based on the input MPL value, horizontal routing tracks are automatically added at positions that are at MPL/2 above and below the y-coordinate of external pins. Without these additional tracks, the pin length can only be specified as a multiple of the default routing grid spacing, potentially leading to unnecessary resource wastage or routing failures. Thus, by using this feature, designers can deliver solutions that meet the desired minimum specifications without wasting resources.

On the other hand, a coarser routing grid can be employed when the layout is relatively simple but the search space is large, in order to effectively reduce the overall design space. For example, cells such as INV and BUF with high driving strength (e.g., BUFx24) often have simple internal connectivity yet feature a wide cell width, resulting in an extensive routable area. In such cases, we propose to remove routing tracks that fall outside the contact coordinates (source, drain, and gate) defined by the default routing grid. This approach helps to significantly reduce runtime while still preserving layout quality.

The two features described above can be optionally enabled by the user.[2] Au-MEDAL allows the routing grid for each metal layer to be customized individually. In addition, the additional routing tracks can be configured to exist in the horizontal, vertical, or both directions, and users can also decide how many times denser the horizontal and vertical grid spacings should be. These options are designed to effectively balance the trade-off between runtime and layout quality.

D. Pin Accessibility–aware Routing

Au-MEDAL proposes two methods to account for pin accessibility. The first is an objective function that maximizes pin extension, and the second guarantees pin accessibility at the block-level routing stage.

[2]To simplify user access, our implementation provides parameters such as `low_resolution_routing` and `complexity_level`. Implementation details are available in our repository [22].

TABLE IV: # of DRVs for the provided standard cells: All ASAP7 reference cells and all our generated cells are completely DRC-clean in all evaluated cases. "Total" differs according to the respective works.

Place	Route	DRV types					# Cells	
		S2S	S2T	T2T	C2C	etc	Clean	Total
ASAP7 [17]	ASAP7	0	0	0	0	0	**172**	**172**
	Ours	0	0	0	0	0	**172**	**172**
Csyn-fp [21]	Csyn-fp	0	22	28	60	10	92	169
	Ours	0	0	0	0	0	**169**	**169**
NCTUcell [15]	NCTUcell	0	1	1	3	3	70	75
	Ours	0	0	0	0	0	**75**	**75**

1) Objective Function for Maximizing Pin Length: As noted in Section II-E, maximizing pin length in standard-cell design can improve a cell's routability. To this end, we enhance the conventional lexicographical optimization by not only minimizing lengths from the highest metal layer down to the lowest, but also by adding prioritization for the pin layer (M1). Specifically, we reorder the optimization sequence so that M1 lengths are progressively minimized beginning at the central horizontal tracks of the cell and then moving outward to the upper and lower tracks. This causes the SMT solver to place metal toward the periphery, naturally creating space for pin extension, reducing optimization complexity, and reducing runtime.

2) Ensuring Pin Accessibility at Block-Level Routing: Even if standard-cell layouts are DRC-clean, DRVs can occur during block-level routing (see Fig. 6(c)). Previous approaches introduced various metrics to improve accessibility but could not fully guarantee DRC-clean block-level routing. To address this, we add SMT constraints that ensure at least one M2 metal is available at each pin's access point. Here, M2 is temporarily inserted during the in-cell routing stage—assuming it is used for block-level routing—and then removed once the final cell layout is generated. With this method, we ensure that every pin has at least one feasible routing scenario.

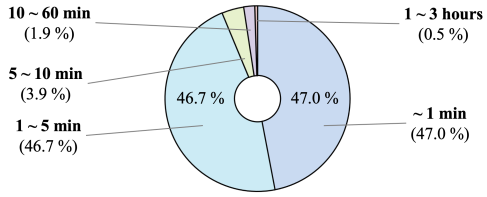


Fig. 8: Runtime statistics of Au-MEDAL for all cells used in the experiments. 93.7% of the cells are routed within five minutes.

IV. Experimental Validations

We evaluate the performance of Au-MEDAL at both the cell and block levels. At the cell level, we analyze standard-cell layout metrics such as the types of DRVs occurring within cells, the number of DRC-clean cells, via count, and metal length. At the block level, we compare power, performance, and area (PPA) and the number of block-level DRVs. Our experiments are conducted on a Linux system with a 2.4 GHz Intel Xeon Gold 6148 processor and 376 GB RAM. The placement input consists of transistor ordering and a CPP at the nm scale, as specified in our design, which determines the x-coordinates; the y-coordinates are determined by the number of fins and the FET type. The routing engine [22] is developed entirely in *Python v3.10*. We use *Z3 v4.15.0* as the SMT solver. We obtain placements from each baseline approach, either from GDS layouts or provided placement data, and perform routing using our engine for comparison. To evaluate Au-MEDAL performance, we generate standard cells based on the ASAP7 PDK. This enables reliable, consistent comparisons and validations of generated standard cells through the application of well-established LVS, DRC, and PEX rule decks. Our evaluation includes ASAP7 [17], Csyn-fp [18], and NCTUcell [15] libraries. SMTCell [3] is excluded from the experimental comparison as it was not originally designed for ASAP7.[3] Since this paper focuses solely on routing, we fix the placements used in previous works and compare only the routing results. ASAP7 reference GDS is obtained from [17]; Csyn-fp layouts are generated directly using [21]; and the GDS files for NCTUcell are obtained directly from the authors of [15].

[3]SMTCell has the following limitations. (1) It does not support bidirectional routing, resulting in the addition of M0 and contact layers that are not part of the ASAP7 process. (2) Contact over active gate, which is not feasible in ASAP7, is enabled in SMTCell by default and cannot be disabled.

TABLE V: Comparison of cell-level power, delay, and Cell-Flex metrics [16]. Each result is evaluated using the same placement from the routed layout of the corresponding baseline [17], [21], [15]. All metrics are normalized to 1; values below 1 indicate improvement. RP and FP refer to rise and fall power, while RD and FD refer to rise and fall delay.

Place	Electrical Charicteristics				Cell-Flex Metrics		
	RP	FP	RD	FD	PAF	HPTA	H2VVA
ASAP7 [17]	**0.976**	**0.991**	**0.999**	**0.999**	1.016	**0.977**	**0.987**
Csyn-fp [21]	1.002	1.025	1.006	1.005	**0.605**	**0.957**	**0.981**
NCTUCell [15]	**0.943**	**0.851**	**0.996**	**0.998**	**0.400**	**0.943**	**0.978**

Au-MEDAL is evaluated on in-cell routing under the following conditions: (1) 7.5T cell track height; (2) LIG and M2 are horizontal, LISD and M1 are bidirectional; (3) the objective function for pin stretchability is enabled; and (4) block-level routing aware routing is disabled except for one case. We apply the block-level routing-aware option only to the FAx1 cell under the ASAP7 placement, where pin accessibility is not guaranteed. This option incurs additional runtime and is therefore omitted for all other cells, as sufficient pin accessibility is achieved without it. All configuration details are provided in our code [22]. All routing results by Au-MEDAL are obtained within 3 hours[4] with 93.7% of the cells being generated in under five minutes (see Fig. 8). We use *Siemens Calibre v23.3* for LVS, DRC, and PEX on all GDS files. We characterize cells with *Synopsys SiliconSmart v23.12*, and extract LEF files using *Cadence Abstract v6.18.0*.

A. Comparison of Cell-level Metrics

We evaluate key metrics of standard cells in three ways. First, we compare the number of DRVs to assess how well each method satisfies the design rules. Second, we compare electrical metrics—such as power and delay—under the same placement. Third, we assess pin accessibility and routing resource availability by adopting the Pin Access Flexibility (PAF) and Track Utilization Flexibility (TUF) metrics proposed in Cell-Flex [16]. PAF measures the average number of routing scenarios that achieve DRC-clean for all of a standard cell's pins at the block level. TUF combines Passing Track (PT)—the number of tracks that span the cell vertically or horizontally—and Via Availability (VA), the number of vias available for block-level routing.

1) Design Rule Violations: Standard-cell layouts must always satisfy all design rules, which is the most fundamental requirement in layout design. We analyze design rule violations in comparison with several previous works. Table IV summarizes the number and types of DRVs in each method. As shown in the table, our method consistently achieves design rule clean layouts. This is due to our effective incorporation of various design rules through metal edge detection logic, inter-layer design rule checking, and off-grid design rule checking.[5]

2) Electrical Characteristics: We compare the electrical characteristics of standard cells (rise/fall power and delay). Table V presents the results obtained by dividing Au-MEDAL's values by the baseline (normalization) and averaging across all cells. A normalized value below 1 indicates that Au-MEDAL outperforms the baseline. The results show that both power and delay metrics are all close to 1, indicating no significant differences. This is because our method focuses solely on routing, while most electrical characteristics are determined by the schematic and transistor placement.

3) Cell-Flex Metrics [16]: We compare PAF and TUF [16]. Here, TUF is split into Horizontal Passing Track Availability (HPTA), representing the availability of M2 tracks, and Horizontal-to-Vertical Via Availability (H2VVA), representing the availability of V2. The PAF, HPTA, and H2VVA in Table V are all normalized to Au-MEDAL,

[4]For the FAx1 cell, which is unique in its high routing complexity, we disabled MOL routing and finer spacing routing grid.

[5]The ACTIVE.LUP.1 design rule is a placement rule related to latch-up, and since it is not clean even in the ASAP7 reference, we exclude only this rule when counting *Calibre*-reported errors in our experiments.

TABLE VI: Comparison of block-level power, performance, area, and number of DRVs. Each result is evaluated using the same placement from the routed layout of the corresponding baseline [17], [21], [15]. Blue indicates improvement, red indicates degradation.

Place	Design	# Cell	Block-level Metric									
			Core Area (um^2)		Total Power (mW)		Effective Clock (ns)		TNS (ns)		# DRVs	
			Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours
ASAP7 [17]	AES	17k	1,782	**1,779** (0.17 %)	12.104	**11.983** (1.00 %)	0.431	**0.428** (0.70 %)	-30.399	**-28.53** (6.15 %)	**88**	**88** (0 %)
	LDPC	52k	6,666	**6,663** (0.05 %)	44.985	**44.753** (0.52 %)	0.754	**0.753** (0.13 %)	-0.025	**-0.021** (16.00 %)	**312**	418 (-33.97 %)
	JPEG	65k	7,577	**7,448** (1.70 %)	81.243	**78.064** (3.98 %)	0.330	**0.323** (2.12 %)	-12.986	**-2.485** (80.86 %)	**336**	340 (-1.14 %)
Csyn-fp [21]	AES	17k	**1,761**	1,797 (-2.04 %)	**11.747**	11.929 (-1.55 %)	**0.420**	0.436 (-3.81 %)	**-26.445**	-30.45 (-15.13 %)	1296	**95** (92.67 %)
	LDPC	52k	**6,590**	6,606 (-0.24 %)	49.298	**49.219** (0.16 %)	0.763	**0.755** (1.05 %)	-0.167	**-0.128** (23.35 %)	5,112	**1,230** (75.94 %)
	JPEG	65k	7,557	**7,244** (4.14 %)	81.037	**77.403** (4.48 %)	0.395	**0.327** (17.22 %)	-48.743	**-8.963** (81.61 %)	9729	**478** (95.09 %)
NCTUcell [15]	AES	17k	1,757	**1,710** (2.68 %)	11.827	**11.588** (2.02 %)	0.449	**0.437** (2.67 %)	-34.497	**-30.11** (12.72 %)	4,118	**88** (97.86 %)
	LDPC	52k	**6,644**	6,655 (-0.17 %)	46.194	**44.815** (2.99 %)	**0.756**	0.758 (-0.26 %)	**-0.553**	-0.756 (-36.71 %)	6,602	**357** (94.59 %)
	JPEG	65k	7,375	**7,260** (1.56 %)	**75.902**	78.076 (-2.86 %)	0.321	**0.317** (1.25 %)	-4.061	**-1.761** (56.64 %)	11,037	**325** (97.06 %)
Average Improvement			-	0.87%	-	1.19%	-	2.34%	-	25.05%	-	57.57 %

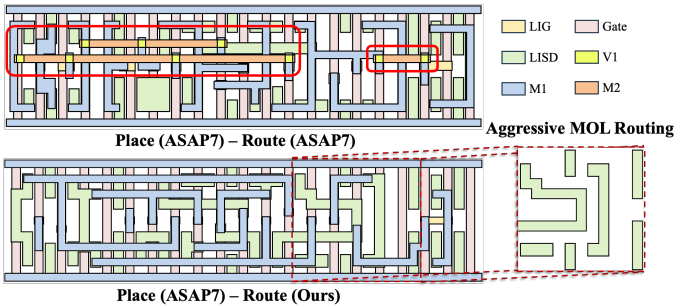


Fig. 9: Layouts of DFFHQNx2: NCTUcell lacks a DFFHQNx2, and while Csyn-fp places it successfully, routing fails.

with any value below 1 indicating that Au-MEDAL outperforms the comparison. First, the PAF results show that the reference ASAP7 exceeds Au-MEDAL by roughly 1–2%. It may be noted that ASAP7's routing was done manually without any fixed grid, and that Au-MEDAL nevertheless achieves pin accessibility comparable to the result of manual routing. Second, against Csyn-fp and NCTUCell, Au-MEDAL's PAF is 1.65× and 2.5× higher, respectively—findings that exhibit a similar trend to the block-level DRC reduction reported in Table VI. HPTA and H2VVA likewise exhibit consistent gains across all benchmarks. This improvement stems from Au-MEDAL's ability to employ a denser routing grid, increasing the use of lower-level metals and vias and thereby reducing reliance on upper-level metals and vias.

In summary, all comparisons are performed under identical placement conditions for fairness, and Au-MEDAL is the only router that consistently produces DRC-clean layouts verified with *Calibre* while delivering strong performance. When compared to the manually designed reference ASAP7 cells, our method reduces the number of cells using M2 metal from 20 to 4 (an 80% reduction). Against Csyn-fp, this figure drops from 19 to 13 (32%), and compared with NCTUcell, from 11 to 3 (73%). Fig. 9 shows an example of a DFFHQNx2 cell routed by Au-MEDAL, highlighting the advantages of our method. While Csyn-fp fails to complete routing, applying our routing engine to its placement results in a 72% reduction in M2 metal usage compared to the ASAP7 reference routing. Notably, applying our engine to the ASAP7 placement yields a routing result that uses no M2 metal at all. Both results are DRC-clean.

4) Flexible Routing Grid Spacing: Au-MEDAL employs a routing grid that is significantly denser than the default. As shown in Fig. 9, this enables it to generate valid solutions—unlike other methods that rely on a fixed grid. In addition, we conduct experiments using a coarser routing grid approach to optimize runtime by eliminating unnecessary grids. This method is effective for large cells with simple net connections but may not be suitable for more complex circuits, where a valid solution might not be found. Accordingly, we apply it to the BUFx24, which consists of 30 gates and only two pins (A and Y). In our approach, routing is performed after removing all grids except those at the net connection points for pins. Both routing quality and runtime are then evaluated. Experimental results show identical routing metrics, with runtime reduced from 2,050 to 480 seconds (77%).

B. Comparison of Block-level Metrics

We evaluate the performance of Au-MEDAL using three block-level designs: AES, LDPC, and JPEG. For these designs, we use only the standard cells commonly available across all comparison methods, utilizing only regular voltage threshold (RVT) cells. Since the reset D flip-flop is not included in the common cell set, we use the ASAP7 reference cell for that flip-flop. Synthesis is performed using *Cadence Genus v21.1*, and P&R is carried out with *Cadence Innovus v21.1*.

AES and JPEG are designed with a utilization of 0.85 and a clock period of 0.300 ns, while LDPC is designed with a utilization of 0.70 and a clock period of 0.750 ns. These design conditions are selected based on the reference ASAP7 [17], targeting points where the worst slack is slightly negative so that each method can produce its fastest possible solution and DRVs are not excessive.

Table VI summarizes the results of the block-level designs. When comparing the improvement ratios over each baseline method, Au-MEDAL shows performance improvements across all metrics (Core Area, Total Power, Effective Clock, TNS, and DRV count) for all methods, except that the number of DRVs increases slightly compared to the manually routed ASAP7. In particular, compared to in-cell routers that are not manually routed, Au-MEDAL achieves an average reduction of 92% in DRV count.

V. Conclusion

We present Au-MEDAL, a new in-cell router for standard-cell layout design. By incorporating metal edge detection—including tips, sides, and corners—Au-MEDAL enables bidirectional routing under extended design rules, achieving cell-level DRC-clean layouts across all evaluated cells. It simultaneously routes Middle-of-Line (MOL) and Back-End-of-Line (BEOL) layers, reducing the number of cell using M2 by 61% on average compared to previous approaches.

Au-MEDAL also maximizes pin stretchability and guarantees at least one valid routing point per pin, significantly reducing block-level DRVs. All layouts are validated to be DRC-clean using *Siemens Calibre v23.3*, and block-level experiments using the ASAP7 7nm PDK demonstrate improvements in area, power, timing, and rule compliance over existing baselines.

Despite these strengths, our framework still has several limitations. To address these, we are extending Au-MEDAL in the following directions. First, we will expand support beyond open-source planar PDKs such as IHP130 [23], GF180 [24], SKY130 [25], and ICSprout55 [26] to include next-generation device architectures such as CFET [2], [19], [20]. Compared to advanced nodes, planar processes offer much greater design freedom, making it even more challenging to achieve manually designed layout quality through grid-based routing. Second, we are implementing a divide-and-conquer approach in which the placement result of a large cell is divided into multiple smaller regions. Routing can be performed in parallel within each region, with the results then merged. This will enable the framework to produce efficient and scalable results even for large-scale cell designs. Third, we seek to design new benchmarks to aid quantitative analysis, assessment and prediction of how improvements in cell-level layout quality translate into block-level PPA benefits.

VI. Acknowledgment

This work is supported by the C-DEN center.

REFERENCES

[1] D. Lee, D. Park, C.-T. Ho, I. Kang, H. Kim and S. Gao, "SP&R: SMT-Based Simultaneous Place-and-Route for Standard Cell Synthesis of Advanced Nodes", *IEEE Trans. CAD* 40(10) (2021), pp. 2142-2155.

[2] C.-K. Cheng, C.-T. Ho, D. Lee, B. Lin and D. Park, "Complementary-FET (CFET) Standard Cell Synthesis Framework for Design and System Technology Co-Optimization Using SMT", *IEEE Trans. VLSI* 29(6), (2021), pp. 1178-1191.

[3] C.-K. Cheng, A. B. Kahng, B. Lin, Y. Wang and D. Yoon, "Gear-Ratio-Aware Standard Cell Layout Framework for DTCO Exploration", *Proc. SLIP*, 2023.

[4] S. Choi, J. Jung, A. B. Kahng, M. Kim, C.-H. Park, B. Pramanik and D. Yoon, "PROBE3.0: A Systematic Framework for Design-Technology Pathfinding with Improved Design Enablement", *IEEE Trans. CAD* 43(4) (2024), pp. 1218-1231.

[5] 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", *Proc. ICCAD*, 2025.

[6] H. Ren and M. Fojtik, "Invited- NVCell: Standard Cell Layout in Advanced Technology Nodes with Reinforcement Learning", *Proc. DAC*, 2021.

[7] C.-T. Ho, A. Ho, M. Fojtik, M. Kim, S. Wei, Y. Li, B. Khailany and H. Ren, "NVCell 2: Routability-Driven Standard Cell Layout in Advanced Nodes with Lattice Graph Routability Model", *Proc. ISPD*, 2023.

[8] J. Yoon and H. Park, "Design-Technology Co-Optimization with Standard Cell Layout Generator for Pin Configurations", *Proc. ISQED*, 2024.

[9] K. Baek and T. Kim, "CSyn-fp: Standard Cell Synthesis of Advanced Nodes With Simultaneous Transistor Folding and Placement", *IEEE Trans. CAD* 43(2) (2024), pp. 627-640.

[10] Y.-L. Li, S.-T. Lin, S. Nishizawa and H. Onodera, "MCell: Multi-Row Cell Layout Synthesis with Resource Constrained MAX-SAT Based Detailed Routing", *Proc. ICCAD*, 2020.

[11] A. C.-W. Liang, C. H.-P. Wen and H.-M. Huang, "A General and Automatic Cell Layout Generation Framework With Implicit Learning on Design Rules", *IEEE Trans. VLSI* 30(9) (2022), pp. 1341-1354.

[12] Y.-L. Li, S.-T. Lin, S. Nishizawa, H.-Y. Su, M.-J. Fong and O. Chen, "NCTUcell: A DDA- and Delay-Aware Cell Library Generator for FinFET Structure With Implicitly Adjustable Grid Map", *IEEE Trans. CAD* 41(12) (2022), pp. 5568-5581.

[13] T.-C. Lee, C.-Y. Yang and Y.-L. Li, "*i*TPlace: Machine Learning-Based Delay-Aware Transistor Placement for Standard Cell Synthesis", *Proc. ICCAD*, 2020.

[14] P. V. Cleeff, S. Hougardy, J. Silvanus and T. Werner, "BonnCell: Automatic Cell Layout in the 7-nm Era", *IEEE Trans. CAD* 39(10) (2020), pp. 2872-2885.

[15] Y.-L. Li, S.-T. Lin, S. Nishizawa, H.-Y. Su, M.-J. Fong and O. Chen, "NCTUcell: A DDA-Aware Cell Library Generator for FinFET Structure with Implicitly Adjustable Grid Map", *Proc. DAC*, 2019.

[16] B. Kang, Y. Yuan, Y. Wang, B. Lin and C.-K. Cheng, "Cell-Flex Metrics for Designing Optimal Standard Cell Layout with Enhanced Cell Layout Flexibility", *Proc. ISPD*, 2025.

[17] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy and G. Yeric, "ASAP7: A 7-nm finFET Predictive Process Design Kit", *Microelectronics Journal* 53 (2016), pp. 105-115.

[18] K. Baek and T. Kim, "Simultaneous Transistor Folding and Placement in Standard Cell Layout Synthesis", *Proc. ICCAD*, 2021.

[19] C.-K. Cheng, C.-T. Ho, D. Lee and D. Park, "A Routability-Driven Complimentary-FET (CFET) Standard Cell Synthesis Framework using SMT", *Proc. ICCAD*, 2020.

[20] T.-W. Lee, T.-X. Lin and Y.-L. Li, "Scalable CFET Cell Library Synthesis With a DRC-aware Lookup Table to Optimize Valid Pin Access", *Proc. ISPD*, 2025.

[21] AutoCellGen (ICCAD-2021).
https://github.com/The-OpenROAD-Project/AutoCellGen

[22] The Au-MEDAL router.
https://github.com/ABKGroup/Au-MEDAL

[23] IHP Open Source PDK.
https://github.com/IHP-GmbH/IHP-Open-PDK

[24] GlobalFoundries GF180MCU Open Source PDK.
https://github.com/google/gf180mcu-pdk

[25] sky130 gdsfactory PDK.
https://github.com/gdsfactory/skywater130

[26] ICSprout55 Open Source PDK.
https://github.com/openecos-projects/ICSprout55-pdk