Multi-Row Standard Cell Layout Synthesis with Enhanced Scalability

Kairong Guo[1,2], Yibo Lin[1,3,4*]
[1]School of Integrated Circuits, Peking University, Beijing, China
[2]School of Integrated Circuit Science and Engineering, Beihang University, Beijing, China
[3]Institute of Electronic Design Automation, Peking University, Wuxi, China
[4]Beijing Advanced Innovation Center for Integrated Circuits, Beijing, China
kr.guo@buaa.edu.cn, yibolin@pku.edu.cn

***Abstract*—Multi-row standard cells are widely adopted in advanced technology nodes, especially for complicated and large cells like multi-bit flip-flops(MBFFs). Due to reduced cell heights and routing tracks, designing standard cell layouts in advanced technology nodes becomes increasingly challenging. Automatic standard cell layout synthesis is being actively explored. However, existing methods face scalability issues when synthesizing large-scale multi-row cells. In this paper, we propose a multi-row cell layout synthesis flow that addresses such scalability issue through a hierarchical approach, including transistor clustering, SMT-based row assignment, transistor-level and cluster-level placement, and genetic-algorithm-based sequential routing, which collectively enables efficient handling of large-scale designs. Experimental results on an industrial 7nm FinFET library demonstrate the ability to handle designs with up to 152 transistors, achieving area reductions up to 23% on large MBFF cells and reducing runtime by up to 36× compared to prior methods.**

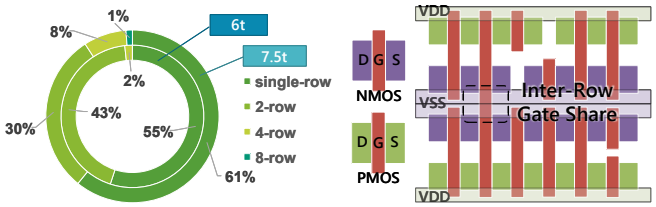
***Index Terms*—standard cell, layout synthesis, transistor-level placement and routing**

I. Introduction

In advanced technology nodes, standard cell libraries have expanded to include a larger variety and number of cells, which makes manual design more time-consuming. This growing complexity necessitates the use of standard cell layout synthesis to automate the design process and improve efficiency. Additionally, the height of standard cells in advanced nodes has been steadily reduced to accommodate increasing transistor densities and enhance chip performance. This reduction in cell height has led to fewer routing tracks per row [1], posing significant challenges to layout design and routability. To address these issues and achieve better performance, the adoption of multi-row standard cells has become increasingly prevalent. As shown in Figure 1(a), we analyzed the cell height distribution of an industrial 7nm FinFET standard cell library, revealing a significant increase in multi-row designs with reduced row heights and fewer routing tracks. Multi-row designs provide greater flexibility in balancing power, performance, and area trade-offs, making them essential in modern standard cell libraries for advanced nodes.

This project is supported in part by Grant QYJS-2023-2303-B and 111 project (B18001).

*Corresponding author.



(a) (b)

Fig. 1: (a) Cell height distribution in an industrial 7nm FinFET library. Almost half of the cells in this library take multiple rows, and the proportion of multi-row cells increases as the number of routing tracks decreases. (b) Sketch of a multi-row cell. NMOS and PMOS are positioned around the power rails.

Standard cell layout synthesis involves transistor-level placement and routing, with the primary objective of minimizing cell area. Most existing studies focus on the layout synthesis of single-row cells. In this scenario, the placement problem is typically formulated as a transistor ordering task, where transistors are flipped and sorted within a single row. Early research in the last century utilizes Euler paths to determine the optimal transistor ordering for generating layouts of combinational logic cells [2] [3]. Other approaches explore branch-and-bound methods to search for transistor ordering [4] [5] or employ simulated annealing algorithms to optimize the ordering [6]. Concurrent routing is a common approach for intra-cell routing. For example, [7] [8] [9] models this process as a SAT/MAXSAT problem. Additionally, SP&R [10] leverages Satisfiability Modulo Theories(SMT) to achieve simultaneous placement and routing, exploring a larger solution space.

Few studies have explored the synthesis of multi-row cells. In this scenario, while the underlying routing problem remains largely the same, placement becomes significantly more complex. The solution space expands as it requires assigning each transistor to an appropriate row while ensuring sufficient global consideration to maximize the benefits of vertical interconnectivity, including inter-row gate sharing, as shown in Figure 1(b). Moreover, multi-row standard cells typically contain a higher number of transistors, further increasing the complexity of the placement process. Studies, such as

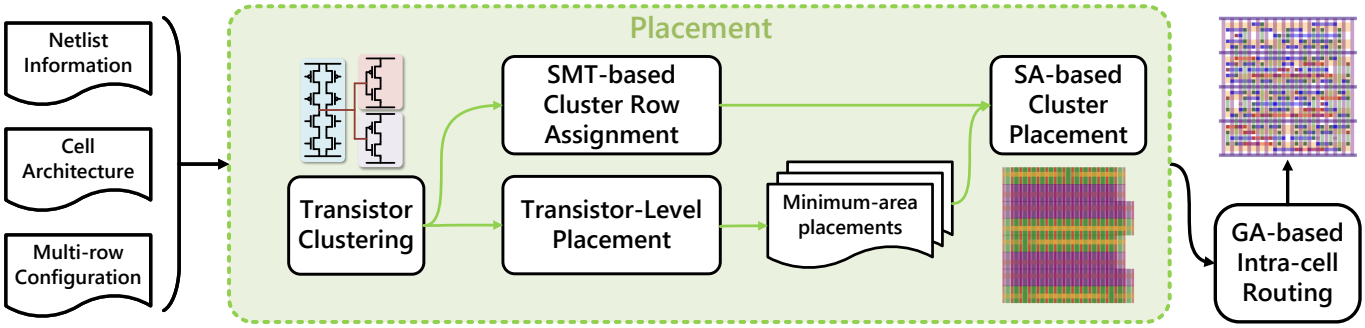


Fig. 2: Proposed multi-row standard cell layout synthesis flow.

BonnCell [4], MCell [8] and SNUCell [5] have made attempts in this direction. However, these approaches lack global consideration and face scalability issues as the design size grows.

To tackle the above challenge, in this work, we propose a multi-row standard cell layout synthesis flow. The key contributions are summarized as follows:

- We propose a multi-row standard cell layout synthesis flow, leveraging a hybrid combinatorial optimization approach to efficiently handle large-scale multi-row cells.
- We introduce a hierarchical approach that first clusters transistors, and then performs placement and SMT-based row assignment for each cluster, followed by a simulated annealing based global optimization step. This method ensures both scalability and global consideration.
- Experimental results on an industrial 7nm FinFET library with 316 cells demonstrate that our approach exhibits excellent scalability, with a maximum area reduction of 23% for large MBFF cells and runtime reductions of up to 36× compared to the recent method [8] for multi-row cells.

The rest of the paper is organized as follows, Section II introduces the motivation; Section III explains the details of the proposed flow; Section IV validates the flow with experimental results; Section V concludes the paper.

II. Preliminaries

A. Global Consideration for Multi-Row Cells

Recent studies specifically addressing the layout synthesis of multi-row standard cells remain limited. Most existing approaches still reduce the problem to a single-row layout scenario for processing. For instance, MCell [8] employs an intuitive algorithm that first generates a single-row transistor placement and then folds it into two or more rows.

BonnCell [4] proposes a method that uses Mixed Integer Programming for row assignment, followed by independent single-row placements. This approach, however, neglects inter-row dependencies during the placement process. Similarly, SNUCell [5] adopts a comparable strategy but leverages Satisfiability Modulo Theories for row assignment instead.

While placement algorithms designed for single-row cells are reasonable for local placement, relying on them to generate multi-row layouts or applying them on a broader scale tends to overlook global considerations, which can result in suboptimal designs.

B. Scalability Challenge

Minimizing the layout area of standard cells is a central task in standard cell layout synthesis, which is known to be an NP-hard optimization problem [11]. Furthermore, existing methods often rely on techniques such as Satisfiability Modulo Theories to pursue optimality, leading to inherent scalability issues.

SP&R [10] discusses the scalability of its approach, validating runtime prediction through tests on 65 combinational logic cells. While promising, this prediction may be overly optimistic, as it primarily focuses on smaller designs and excludes larger ones, such as MBFF. Notably, the largest design presented in [10] comprises only 36 FETs (SDFFSNQ_X1), requiring 6168 seconds for synthesis.

In contrast, NVCell [6] validates its method on designs from an industrial library, claiming the ability to handle up to 114 FETs. To improve runtime, the authors introduced a Transformer Model-Based Clustering Method to accelerate the placement process [12]. Despite these efforts, the runtime remains in the order of tens of thousands of seconds for designs exceeding 80 transistors. Nevertheless, their staged placement approach provides valuable insights, demonstrating that a phased methodology can effectively enhance scalability.

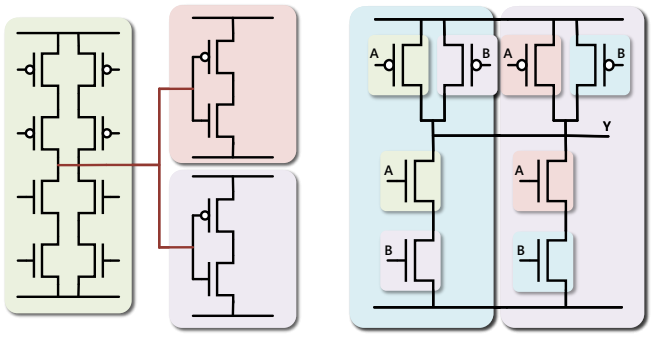
C. Problem Formulation for Multi-Row Cell Layout Synthesis

Given the transistor-level netlist of a standard cell and the specified number of rows for placement, the goal is to generate a standard cell layout optimized for area and wirelength. This involves placing the standard cells into the specified number of rows, minimizing the area of the layout while simultaneously optimizing the total wirelength, including considerations of the benefits of vertical interconnectivity.

III. Algorithm

A. Overview of Proposed Flow

Figure 2 illustrates the proposed multi-row standard cell layout synthesis flow. The synthesis process begins by clustering transistors based on their connectivity, with constraints imposed on the total number of clusters and the number of



(a) (b)

Fig. 3: (a) Clustering rules for sequential logic cells, where transistors within the same cluster are connected only through non-VDD/VSS drains or sources. The red net is split due to its connection to gates. (b) Clustering rules for combinational logic cells, based on branches or gate inputs.

transistors within each cluster. Subsequently, SMT is employed to perform row assignment for each cluster, while a local single-row placement is executed within each cluster. The minimum-area placement of each cluster is obtained by traversing Euler paths. Simulated annealing is utilized to optimize the arrangement of clusters within their respective rows, resulting in the final placement of the cell. The final layout is obtained by applying a maze router that incorporates a genetic algorithm for rip-up and reroute.

The initial transistor clustering significantly reduces the problem size for subsequent row assignment and local placement, thereby substantially decreasing the overall runtime. Simulated annealing is employed for a global refinement of the placement. A maze router based on genetic algorithms is employed for routing, leveraging its capability for parallel searches to enhance routing efficiency while producing a diverse set of routing solutions.

B. Transistor Clustering

We perform transistor clustering to handle larger sequential logic cells, such as MBFFs. As illustrated in Fig. 3, clustering is based on the connectivity of transistors. Specifically, transistors within the same cluster are connected only through nets that are neither VDD nor VSS, and these nets are exclusively connected to the drains or sources of the transistors.

When the source or drain of horizontally adjacent transistors are connected to the same net, they can overlap, enabling diffusion sharing to reduce the overall area. This clustering approach ensures that transistors within the same cluster are functionally related and capable of achieving diffusion sharing among themselves. On the other hand, transistors belonging to different clusters can only achieve diffusion sharing through VDD or VSS.

For combinational logic cells, the clustering process typically results in a single cluster under the aforementioned rules. However, to adapt to this flow, these cells are clustered based on branches or gate inputs. This is because larger combinational logic cells often exhibit highly repetitive structures.

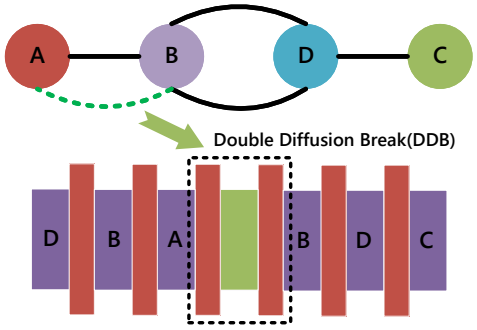


Fig. 4: Example of inserting a double diffusion break, which can also be inserted between another pair of odd-degree nodes.

C. Transistor-Level Placement and Row Assignment

After performing transistor clustering, we subsequently generate the placement for each transistor within individual clusters and allocate rows for each cluster. These two parts are included in the same section because both make use of Euler paths, but in different ways. The former obtains minimum-area placements by traversing Euler paths, while the latter predicts the width of each row using Euler paths. Also, after clustering, these two processes can run in parallel without a specific order.

If gates are treated as edges and source/drain terminals as vertices, then an Euler path on the resulting graph corresponds to the minimum-area transistor chain [2]. By performing this operation separately for NMOS and PMOS transistors, and then merging the resulting chains, we obtain the minimum-area transistor placement.

It is important to note that a necessary condition for an Euler path to exist is that at most two vertices have an odd degree. Since odd-degree vertices appear in pairs, an Euler path can be ensured by inserting additional edges, referred to as a diffusion break. The library we used for testing employs a double diffusion break (DDB), as shown in Figure 4. The DDB can also be viewed as two dummy transistors.

The notations used in this section can be found in Table I.

TABLE I: Notations for Section III-C

Term	Description
$C_{i,n}, R_{i,p}$	Set of NMOS/PMOS in i^{th} cluster/row
G_T	Graph constituted by transistors from set T
$ODD(G)$	Set of odd-degree vertices in Graph G
$odd(G)$	Number of odd-degree vertices in Graph G
$r(t), r(C)$	The row number of transistor t/cluster C
N	Set of multi-pin nets
$n, p, y(p)$	Net n and pin p, and the y-coordinate of pin p
$w_i, w(t)$	Width of i^{th} row and width of transistor t
DDB	Width of double diffusion break

1) Traversing Euler Paths: Enumerating all Euler paths on an undirected graph involves exponential computational complexity as the size of the graph grows. However, when the starting point of the path is fixed and the graph size is limited, the computational effort becomes manageable. The determination of the path starting point and the dummy

insertion method are described as follows. For a cluster C, taking the graph G_{C_n} as an example:

- If $odd(G_{C_n}) = 0$, the vertex corresponding to VSS is selected as the starting point, because transistors belonging to different clusters can only achieve diffusion sharing through VDD or VSS.
- If $odd(G_{C_n}) = 2$, two vertices from $ODD(G_{C_n})$ are chosen as the starting points.
- If $odd(G_{C_n}) > 2$, a different pair of vertices is selected from $ODD(G_{C_n})$ as the starting points, and the remaining vertices in $ODD(G_{C_n})$ are paired and connected. Then, the Euler path traversal begins. This is more complex as it requires traversing through different starting points and connection configurations of the remaining vertices in $ODD(G_{C_n})$. Fortunately, experimental results show that typically $odd(G_{C_n}) \leq 4$.

The specific Euler Paths traversal method, after selecting a pair of starting points for G_{C_n} and G_{C_p}, is to perform a depth-first search (DFS) simultaneously on both graphs using point pairs. The routability-optimal result is selected based on gate sharing as the final outcome.

2) SMT-based Cluster Row Assignment: We assume that the transistors within the same cluster are functionally related. Given that the size of each cluster is relatively small, the local nets can be effectively handled with horizontal interconnects using metal layer M0, without the need for multi-row designs. Therefore, placing the entire cluster within a single row is a reasonable approach. This method shifts the focus of row assignment from individual transistors to entire clusters, which significantly reduces the problem scale. While this approach may lack flexibility for small cells, it is well-suited for large cells.

The specific formulation for the SMT-based Cluster Row Assignment is as follows. Given all transistors T, the transistor netlist N, and the number of rows k to be assigned, the objective is to lexicographically optimize the overall cell width and vertical interconnects:

$$\min_{r(t),t\in T} \quad \begin{cases} \max\limits_{0\leq i\leq k} w_i, \\ \sum\limits_{n\in N} \max\limits_{p\in n} y(p) - \min\limits_{p\in n} y(p) \end{cases} \qquad (1)$$

$$\text{subject to} \quad \begin{cases} 0 \leq r(C) \leq k, \quad \forall C \subseteq T, \\ r(t) = r(C), \quad \forall t \in C \end{cases}$$

w_i is estimated from the Euler path as follows:

$$w_i = \max(w_{i,n}, w_{i,p})$$
$$w_{i,n/p} = \sum_{t\in R_{i,n/p}} w(t) + DDB \times (\frac{\text{odd}(G_{R_{i,n/p}})}{2} - 1) \qquad (2)$$

We use the SMT solver Z3 [13] to solve the optimization problem, with a maximum search time limit. If the time limit is exceeded, the current best solution is returned.

D. Cluster-Level Placement

After obtaining the placement and row assignment for each cluster, we employ a simulated annealing algorithm to determine the exact position and orientation of each cluster within its assigned row. The optimization objective combines the overall cell area, inter-row gate sharing, and weighted half-perimeter wirelength (HPWL).

While each cluster is confined to its designated row, this approach incorporates global considerations for the multi-row placement. The use of simulated annealing is due to its ease of implementation and the manageable scale of the Cluster-Level Placement.

E. Genetic Algorithms based Sequential Routing

For the routing part, we employ a sequential routing approach, using a rip-up and reroute strategy based on the positional information, to explore routing solutions. We adopt a Genetic Algorithm framework for search, with a routing segment-based genetic encoding strategy which was already used in channel routing in the previous century [14]. At the core, a maze router based on the A* algorithm is used to generate a large number of initial routing results and perform rerouting. The details are as follows.

1) A based Maze Router with Design Rule Checker:* The generation of initial solutions and rerouting are both handled by an A* based Maze Router, which performs routing on a grid space determined by the routing tracks. The pins of unrouted nets are dynamically selected based on the placement information and the routing of other nets. The given position of IO pins serves as a soft constraint, guiding the routing process. During the A* searching, the router continuously checks the design rules in real-time based on the occupancy status of the grid space.

2) Genetic Algorithms Framework: The framework for the rip-up and reroute search using Genetic Algorithm is shown in Algorithm 1. The generation of the initial solution and the rerouting process are both based on a random routing order.

Algorithm 1 Genetic Algorithm Framework

Input: Initial routing solutions $S = \{\mathbb{R}_1, \mathbb{R}_2, \mathbb{R}_3, \ldots, \mathbb{R}_k\}$, generations G
for $g = 1$ **to** G **do**
  New generation $S' \leftarrow \emptyset$
  for $i = 1$ **to** $\frac{k}{2}$ **do**
    Select $(\mathbb{R}_i, \mathbb{R}_j)$ with roulette wheel selection
    $(\mathbb{R}'_i, \mathbb{R}'_j) \leftarrow$ crossover$(\mathbb{R}_i, \mathbb{R}_j)$
    $\mathbb{R}'_i \leftarrow$ mutate$(\mathbb{R}'_i)$, $\mathbb{R}'_j \leftarrow$ mutate$(\mathbb{R}'_j)$
    $S' \leftarrow S' \cup \mathbb{R}'_i, \mathbb{R}'_j$
  end for
  $S \leftarrow$ Sort&SelectTopkFitness$(S \cup S')$
  if The solution with the highest fitness ($\mathbb{R}_0 \in S$) is completed **then**
    Output $\mathbb{R}_0$ and terminate
  end if
end for
Output $\mathbb{R}_0$

TABLE II: Comparison with MCell [8]. The metrics include area, metal length (ML), via count, increased m2 track usage, and runtime (measured in seconds), where runtime P and runtime R represent runtime of placement and routing respectively.

Cell	Size	MCell [8]							Our				
		area	ML	m2	via	runtime P	runtime R	runtime	area	ML	m2	via	runtime
DFFQA_X0.7(2bit)	52	0.94	0.71	0.32	0.84	1.93	1.32	3.25	0.94	0.81	0.27	0.91	2.60
DFFQA_X1.3(2bit)	56	0.94	0.82	0.54	0.89	32.52	1.45	33.97	0.89	0.70	0.18	0.81	1.96
DFFQA_X2(2bit)	56	0.77	0.70	0.50	0.77	34.22	1.37	35.59	0.82	0.51	0.02	0.58	1.74
DFFQNA_X0.7(2bit)	52	0.94	0.75	0.23	0.84	4.95	2.12	7.06	0.94	0.92	0.51	1.07	4.65
DFFQNA_X1.3(2bit)	56	0.94	0.81	0.53	0.95	157.94	1.41	159.35	0.89	0.59	0.02	0.65	1.98
DFFQNA_X2(2bit)	56	0.77	0.74	0.43	0.97	168.91	1.31	170.22	0.82	0.53	0.02	0.60	2.17
INV_X8F	16	1.00	0.59	0.03	0.64	331.44	0.35	331.79	1.00	0.54	0.02	0.49	3.36
INV_X8N	16	1.00	0.54	0.02	0.49	331.69	0.37	332.06	1.00	0.54	0.02	0.49	3.49
NAND2_X4N	16	1.00	0.74	0.10	0.92	37.88	0.42	38.30	1.00	0.57	0.03	0.64	3.97
NAND2_X4R	16	1.00	0.74	0.10	0.92	41.63	0.45	42.07	1.00	0.57	0.03	0.64	3.88
NOR2_X4F	16	1.00	0.77	0.14	1.00	41.31	0.43	41.74	1.00	0.57	0.03	0.69	3.79
NOR2_X4N	16	1.00	0.77	0.14	1.00	42.85	0.44	43.29	1.00	0.57	0.03	0.69	3.98
Average		0.94	0.72	0.27	0.84	107.67	1.00	108.67	0.94	**0.62**	**0.11**	**0.69**	**3.05**

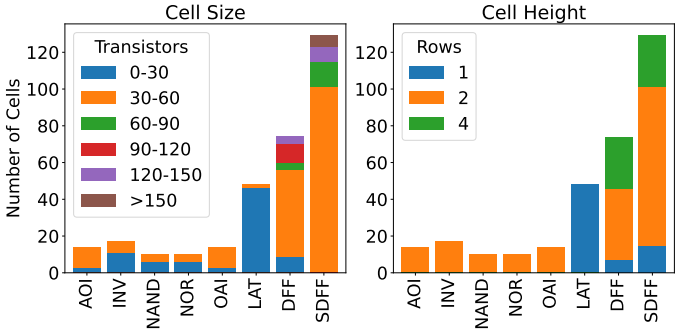


Fig. 5: The left figure shows the sizes of selected cells, while the right figure illustrates the original row configurations of these cells. Selection Criterion:>30 transistors or multi-row.

We use an R-tree to record the position information of each routed net, which is then used for the crossover and mutation operations:

- **crossover**$(\mathbb{R}_i, \mathbb{R}_j)$**:**A random x or y coordinate is selected.For example, when selecting the x-coordinate, all routed nets in $\mathbb{R}_i$ that lie to the left of x are merged with all routed nets in $\mathbb{R}_j$ that lie to the right of x. A reroute is then performed to obtain $\mathbb{R}'_i$. The generation of $\mathbb{R}'_j$ follows the same procedure.
- **mutate**$(\mathbb{R}_i)$**:**A random rectangular region is selected, and all routed nets in $\mathbb{R}_i$ that overlap with this region are removed. A reroute is then performed to obtain $\mathbb{R}'_i$.

The fitness is determined by the proportion of routed nets and the HPWL.

IV. Experimental Results

Our work is implemented in C++ and executed on a PC equipped with an Intel Ultra 5 125H CPU, utilizing up to 4 threads for the GA-based routing. In this study, we select 316 standard cell designs from an industrial standard cell library at a 7nm FinFET technology node. As a cell should be large enough to be designed as multi-row, we select cells with more than 30 transistors or whose original manual implementations are multi-row. Additionally, to ensure the comprehensiveness of the cell types, we also include latches from the industrial library even if they do not meet the above criteria. The statistics about selected cell types and their original row counts are plotted in Figure 5. These cells span a wide range of functionalities, including various types of flip-flops and combinational logic gates. Their sizes range from 16 transistors to 152 transistors, reflecting the diverse design requirements encountered in practical applications. The experiments are organized as follows:

- **Exp. 1: Efficiency Comparison:** We compare our algorithm with MCell [8] to validate the performance our placement.
- **Exp. 2: Scalability Validation:** We generate all 316 cells as 2-row, 2-bit DFF/SDFF cells as 3-row, and 2-bit and 4-bit DFF/SDFF cells as 4-row to validate the scalability of our algorithm.

All the subsequently presented metrics, including area, wirelength, are normalized against manually designed layouts from the industrial library.

A. Efficiency Comparison

We obtained the executable file for MCell [8], used it for placement, and then applied our router for routing. The results are compared with those generated by our own flow. Since MCell is based on ASAP7 [15], it has compatibility issues when adapting to design rules from actual industrial libraries. Thus, we compare with a subset of cells where MCell can successfully complete, as shown in Table II. Our algorithm achieves a 36× runtime speedup with a 13.9% reduction in metal length and 17.9% reduction in via usage, while maintaining the same area ratio on average. The speedup is primarily observed in the placement phase, as the routing time is negligible compared to the placement time in MCell. This indicates that the hierarchical placement approach substantially enhances cell synthesis efficiency.

B. Scalability Validation

We generated all 316 cells as 2-row designs, with over 90% of them being successfully routable and LVS/DRC clean. The results are summarized in Table III, demonstrating improvements in area, metal length, and via usage compared to

TABLE III: Generation results of all cells. The results include designs with 2-row, 3-row, and 4-row configurations.

Cell Type	Suc/Tot	Size	Avg. RT	Area	ML	Via	m2 usage	Suc Rate
2-Row								
AOI	14/14	24-36	36.49	1.00	0.86	0.73	0.04	
INV	17/17	16-40	10.93	1.00	0.64	0.69	0.01	
LAT	48/48	16-30	0.77	1.32	0.93	0.87	-0.03	
NAND	10/10	16-36	9.08	1.00	0.77	0.66	0.02	
NOR	10/10	16-36	9.14	1.00	0.78	0.68	0.02	
OAI	14/14	24-36	28.70	1.00	0.86	0.73	0.06	
DFF(1,2,4B)	66/74	26-120	390.73	0.93	0.84	0.91	0.16	
SDFF(1,2,4B)	117/129	34-152	373.28	0.95	1.02	1.04	0.33	
Overall Averages/Sum(m2)				0.98	0.85	0.86	1919	**90.25%**
3-Row								
DFF(2B)	14/14	52-60	38.14	0.90	0.82	0.98	0.27	
SDFF(2B)	14/14	68-78	191.3	0.92	1.01	1.11	0.63	
Overall Averages/Sum(m2)				0.91	0.84	0.96	2440	
4-Row								
DFF(2,4B)	21/28	52-120	343.11	1.03	0.83	0.97	0.26	
SDFF(2,4B)	16/28	68-152	860.79	1.04	1.03	1.13	0.65	
Overall Averages/Sum(m2)				1.03	0.83	0.95	2445	

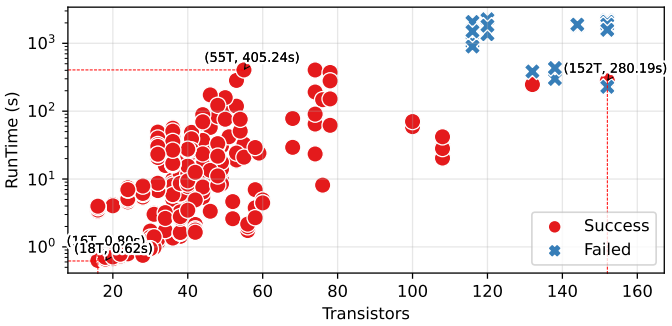


Fig. 6: The outcome and runtime of generating all cells with a 2-row configuration. The horizontal axis represents the number of transistors(T for short) in each cell.

manually designed layouts. Notably, as illustrated in Figure 6, all cases with fewer than 100 transistors were successfully routed, and the runtime for all successfully routed cells was on the order of hundreds of seconds. The largest cell successfully routed contains 152 transistors. Additionally, for cells originally designed as multi-row, 66.4% of the cells achieved equal or smaller areas. As shown in Figure 7, our flow demonstrates effective area optimization, particularly for large MBFFs, with a maximum area reduction of up to 23%.

Furthermore, we applied our approach to more complex cells with greater row counts (beyond two rows). As shown in Table III, the runtime remains efficient even as the complexity increases, demonstrating the scalability of our method.

We also observe that our flow performs less favorably in terms of area for smaller designs originally implemented as single-row cells, due to insufficient flexibility in handling such designs. For example, all latch cells in the original design are single-row and relatively small (<30T), and the layout generated by our flow has a larger area. We will optimize the performance for these cells in the future.

V. Conclusion

In this paper, we presented a novel multi-row standard cell layout synthesis flow that significantly improves the scalability

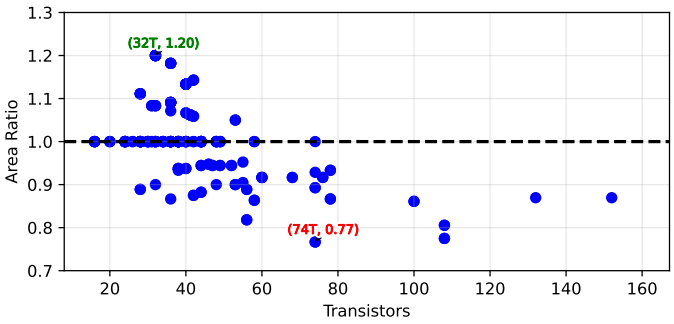


Fig. 7: Area comparison with manual designs (normalized).

of designing large-scale multi-row cells. By leveraging a hierarchical approach that employs hybrid combinatorial optimization, our method successfully addresses the challenges inherent in the synthesis of large multi-row standard cells. Experimental results on an industrial 7nm FinFET library with 316 cells demonstrate that our approach exhibits excellent scalability, achieving a maximum area reduction of 23% for large MBFF cells and delivering runtime reductions of up to 36× compared to the recent method [8] for multi-row cells. Furthermore, the fact that the transistor-level placement step in our flow can be replaced with other single-row placement algorithms, and the parallelization of placement for each cluster, shows both the compatibility and scalability of the flow. These results highlight the potential of our approach for advancing multi-row standard cell design. Future work will focus on optimizing routability during placement and improving performance on single-row and relatively small cells.

References

[1] J. Ryckaert, M. H. Na, P. Weckx, D. Jang, P. Schuddinck, B. Chehab, S. Patli, S. Sarkar, O. Zografos, R. Baert *et al.*, "Enabling sub-5nm cmos technology scaling thinner and taller!" in *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2019, pp. 29–4.

[2] R. L. Maiasz and J. P. Hayes, "Layout optimization of CMOS functional cells," in *24th ACM/IEEE conference proceedings on Design automation conference - DAC '87*. Miami Beach, Florida, United States: ACM Press, 1987, pp. 544–551.

[3] M. Lefebvre and C. Chan, "Optimal ordering of gate signals in CMOS complex gates," in *1989 Proceedings of the IEEE Custom Integrated Circuits Conference*. San Diego, CA, USA: IEEE, 1989, pp. 17.5/1–17.5/4.

[4] P. Van Cleeff, S. Hougardy, J. Silvanus, and T. Werner, "BonnCell: Automatic Cell Layout in the 7-nm Era," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2872–2885, 2020.

[5] S. Chung, H. Seo, H. Cho, K. Choi, and T. Kim, "Optimal Layout Synthesis of Multi-Row Standard Cells for Advanced Technology Nodes," pp. 1–8, 2024.

[6] 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," in *Proceedings of the 2023 International Symposium on Physical Design*. Virtual Event USA: ACM, 2023, pp. 44–52.

[7] D. Park, I. Kang, Y. Kim, S. Gao, B. Lin, and C.-K. Cheng, "ROAD: Routability Analysis and Diagnosis Framework Based on SAT Techniques," in *Proceedings of the 2019 International Symposium on Physical Design*. San Francisco CA USA: ACM, 2019, pp. 65–72.

[8] 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," in *Proceedings of the 39th International Conference on Computer-Aided Design*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–8.

[9] J.-C. Tsai, W.-M. Hsu, Y.-T. Hsieh, Y.-J. Li, W. Huang, C. Ho, J.-H. Yang, and H.-L. Huang, "MAXCell: PPA-Directed Multi-Height Cell Layout Routing Optimization using Anytime MaXSAT with Constraint Learning," *New York*, 2024.

[10] D. Lee, D. Park, C.-T. Ho, I. Kang, H. Kim, S. Gao, B. Lin, and C.-K. Cheng, "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*, vol. 40, no. 10, pp. 2142–2155, 2021.

[11] S. Chakravarty, X. He, and S. Ravi, "Minimum area layout of series-parallel transistor networks is np-hard," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 10, no. 7, pp. 943–949, 1991.

[12] C.-T. Ho, A. Chandna, D. Guan, A. Ho, M. Kim, Y. Li, and H. Ren, "Novel transformer model based clustering method for standard cell design automation," in *Proceedings of the 2024 International Symposium on Physical Design*, 2024, pp. 195–203.

[13] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340.

[14] J. Lienig, "A parallel genetic algorithm for performance-driven VLSI routing," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 29–39, 1997.

[15] 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*, vol. 53, pp. 105–115, 2016.