# Migrating Standard Cells for Multiple Drive Strengths by Routing Imitation

Xiaohan Gao
*School of Computer Science, Peking University*
*School of Integrated Circuits, Peking University*
xiaohangao@pku.edu.cn

Haoyi Zhang
*School of Integrated Circuits, Peking University*
hy.zhang@pku.edu.cn

Zhu Pan
*Primarius Technology*
panzhu@primarius-tech.com

Yibo Lin[*]
*School of Integrated Circuits, Peking University*
*Beijing Advanced Innovation Center for Integrated Circuits*
*Institute of Electronic Design Automation, Wuxi*
yibolin@pku.edu.cn

Runsheng Wang
*School of Integrated Circuits, Peking University*
*Beijing Advanced Innovation Center for Integrated Circuits*
*Institute of Electronic Design Automation, Wuxi*
r.wang@pku.edu.cn

Ru Huang
*School of Integrated Circuits, Peking University*
*Beijing Advanced Innovation Center for Integrated Circuits*
*Institute of Electronic Design Automation, Wuxi*
ruhuang@pku.edu.cn

***Abstract*—Standard cells are critical primitives of modern integrated circuits. Designing standard cells requires time-consuming manual optimization. Within a standard cell library, designers resize standard cells with the same functionality for different drive strengths. Building up all the layouts of all drive strengths from scratch introduces a lot of repetitive and redundant work, especially in routing. We propose a standard cell resizing framework to migrate a layout to another drive strength by imitating the routing of the existing layout. Experimental results demonstrate that our framework is capable of synthesizing layouts with competitive performance with the manual layouts on an industrial standard cell library.**

***Index Terms*—standard cell layout, routing, point cloud**

## I. Introduction

Standard cells are the cornerstone of the subsequent flow for digital integrated circuits (ICs). As each standard cell is reused millions of times as a basic unit for ICs, it always requires considerable effort to optimize the design of a standard cell. The procedure of designing cell layouts is a tough task and remains hand-crafted in most standard cell libraries. Such a procedure is very tedious, especially when designers manually finish the routing under limited routing space and complex rules. As layouts are critical to the area, delay, and power of standard cells, cell routing needs to be fast and high-quality to enable quick exploration of different topologies and improve the eventual performance.

Modern standard cell libraries typically contain around 100 to 2000 standard cells, and some downstream applications even request customized cells that are not included in the original library [1]. Meanwhile, a standard cell library consists of cells with diverse functionalities, including combinational logic units (such as `AND`), and sequential units (such as `Latch` and D-flipflop `DFF`). The exploding number of cells for a library mainly comes from versatile implementations of functionalities. Drive strength is one major factor that introduces variations to the implementations, and a standard cell library obtains cells of multiple drive strengths by resizing the schematic. For example, an industrial customized standard cell library

[*] Corresponding author.

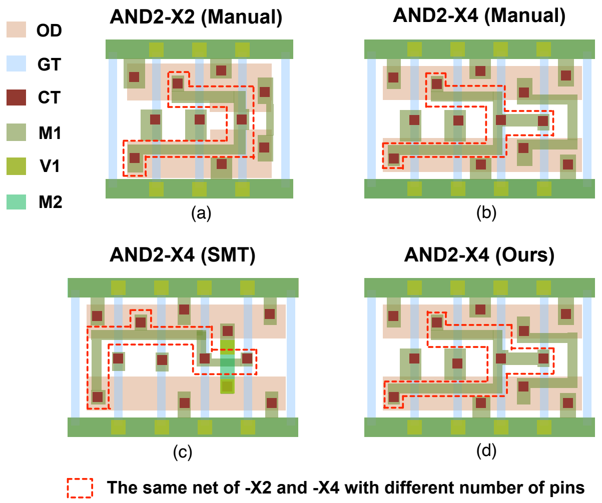

Fig. 1: A visual comparison: the `AND2-X2` (manual) shares patterns with `AND2-X4` (manual); our framework generates a very similar layout for `AND2-X4` after seeing the layout of `AND2-X2`.

of SMIC28 PDK has 17 buffers (`BUFX1`, ···, `BUFX40`) and 5 scan D-FlipFlop (`SDQX1`, ···, `SDQX6`). We observe that there exists geometrical homogeneity and shared patterns among the manual layouts for a series of cells with different drive strengths.

Existing approaches for standard cell layouts pay little attention to the geometrical similarity between cells with different drive strengths. Previous work utilizes Boolean satisfiability (SAT) to find a feasible routing solution [2]–[4]. Further research proposes ILP/MILP-based placement [5], [6] and routing [7]–[10], or simultaneously solving placement and routing [11]. However, the SAT, SMT, or ILP-based methods suffer from time explosion when the number of variables scales up with cell complexity. Other studies approximate the placement and routing solutions with routing planning [12], greedy heuristics [13], graph-based search [14], [15], and machine learning [16]–[18]. However, those studies synthesize each cell independently and ignore reusing layout topologies. Another line of standard cell layout research considers layout migration based on layout compaction from a previous technology node to a newer technology node [19]–[24]. Layout compaction is limited to the case that there exists a one-to-one correspondence between every metal segment of the layouts before and after migration. However, under the scenario of resizing standard cells for multiple drive strengths, the number of pins and the number of wire segments can change a lot.

In order to take advantage of the geometrical homogeneity between cells with different drive strengths, which has been neglected all along, we propose the problem of *routing imitation*. *Routing imitation* indicates transferring an existing routing to another cell with a schematic-level-similar topology. Figure 1 (a) and Figure 1 (b) show that the manual layouts of `AND2-X2` and `AND2-X4` are similar in geometrical topology. However, as shown in Figure 1 (c), an SMT-based solver [11] cannot guarantee such topological similarity when the numbers of pins and wire segments are changed. Our *routing imitation* approach follows the guidance of layout `AND-X2` and generates a topologically similar solution shown in Figure 1 (d). Even for the same net with a different number of pins, the *routing imitation* captures accurate topology information and is visibly similar to the manual layout. The layout generated by *routing imitation* achieves competitive performance after standard cell characterization. *Routing imitation* benefits the layout synthesis from two perspectives, i.e., solution quality, and search efficiency: (1) It is not practical to model all the performance requirements as constraints. Imitating the manually optimized routing implicitly models the requirements as following the guidance of an optimized layout, which is more likely to generate high-quality layouts. (2) Imitating an existing routing dramatically prunes the search space and starts the search process near feasible local optima.

This problem of *routing imitation* is posed by a leading IP design team in industry, as manually drawing the routing of cells with similar topologies is extremely tedious and time-consuming. This paper presents a layout synthesis framework for standard cell *routing imitation*. We highlight our main contributions as follows:

- We explore the geometrical homogeneity of the cell layouts for multiple drive strengths, which reveals the potential for reusing layouts inside a standard cell library.
- We propose the concept of *routing imitation* which is a new paradigm of generating layout by migration.
- We propose novel techniques based on point cloud to abstract and migrate the layout topology with a complete flow to realize the final layout.
- Experimental results demonstrate that our framework can produce high-quality cell layouts in seconds and achieve competitive geometrical and electrical performance compared with manual layouts in an industrial customized standard cell library.

The rest of this paper is organized as follows. Section II defines our problem formulation. Section III gives an overview of our layout synthesis framework and provides detailed algorithms. Section IV presents the experimental setup and results, and Section V concludes this paper.

## II. Preliminaries

In this section, we formulate our *routing imitation* problem and briefly introduce how we abstract the layout topology for the *routing imitation*.

### A. Problem Formulation

We formulate the *routing imitation* problem as follows:

**Problem 1** (Routing Imitation between Cells with Different Drive Strengths)**.** Given a layout of some standard cell at drive strength $m$, the same functional cell at another drive strength $k$ (without layout), and a set of design rules. Regard the cell with drive strength $m$ as the guidance cell, and the cell with drive strength $k$ as the target cell. The goal is to migrate the layout of the guidance cell to synthesize the layout for the target cell, without violating the design rules and with performance close to manual results.

### B. Represent Pins as Point Cloud

The precondition of migrating layout topology is a representation of the layout topology. Our framework presents an abstraction for pin locations of the standard cell layout using point cloud.

A point cloud is a collection of points in 3D space with their coordinates and other properties. Our framework considers the pin locations as a point cloud of 3D coordinates $\{(x_p, y_p, l_p)\}$ in which the $x_p, y_p$ represents the 2D coordinate and $l_p$ represents the metal layer.

Point cloud representation introduces a geometrical transformation for the layout migration. For two cells $g$ and $t$, the migration of pin locations from $g$ to $t$ derives an alignment between the point cloud $\{x_p^g, y_p^g, l_p^g\}$ and $\{x_p^t, y_p^t, l_p^t\}$. Point cloud registration is the process of aligning two point clouds. The iterative closest point algorithm is a widely used technique to iteratively find the alignment and the geometrical transform between two point clouds [25]. Our framework adopts the technique to characterize the geometrical transformation of the migration process.

## III. Algorithm

In this section, we introduce the detailed algorithms. Figure 2 illustrates the framework of our approach that synthesizes the layout of a standard cell from a guidance layout with another drive strength. Our framework extracts the placement and routing information with LVS from the guidance layout. The placement information is represented with the relative position of transistors, and the routing information is represented with the Steiner tree. We preserve the relative position of transistors and directly synthesize the placement over the ordered sequence of transistors for the target cell using an industrial tool. Our framework figures out a geometrical transformation from the guidance layout to the target layout. Then we apply the geometrical transformation to the Steiner trees of the guidance layout to migrate routing topology, which generates rough Steiner trees for nets of the target cell. To get violation-free routing paths, our framework plans for via selection and metal shapes based on the generated Steiner trees. The framework constrains a grid-based routing with the well-planned Steiner trees to complete the layout synthesis.

### A. Topology Migration

To make the most of the guidance layout, our framework proposes an algorithm that migrates the structural routing information (i.e., routing topology represented with Steiner trees) to a new layout. The algorithm studies the structural routing information from two perspectives: the geometrical position of the Steiner points and the topological connection relationship between the Steiner points. Therefore, the algorithm first calculates a geometrical transformation from the pin positions of the guidance layout to pins of the target layout. The geometrical transformation is applied to generate positions of the Steiner points and vias on the target layout. Then the algorithm

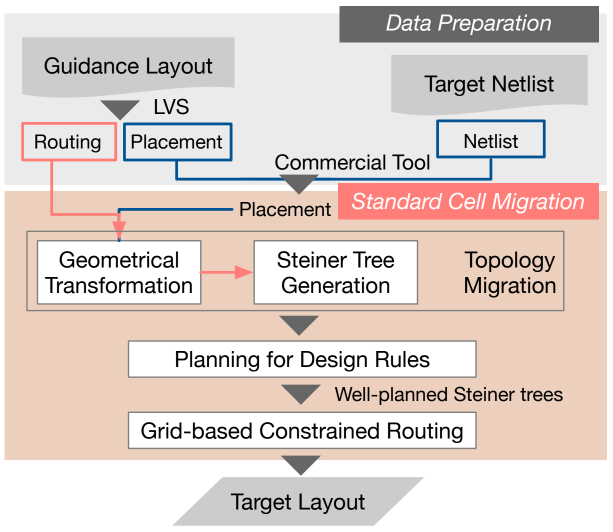

Fig. 2: The overall flow of our framework.

organizes the generated Steiner points and vias to compose a Steiner tree for each net of the target cell.

*1) Calculate geometrical transformation:* Note that the placement of the target cell preserves the relative positions of the transistors compared to the guidance layout. Thus the pin positions of the target cell have a similar distribution to the pin positions of the guidance layout. We consider the pin positions as a set of points in space, and the problem turns into finding a geometrical transformation from a set of points to another set of points. Figure 3 shows how our algorithm tackles this problem. First, our algorithm formulates the pin positions as a point cloud. The point cloud $C_g$ and point cloud $C_t$ represent the pin positions of the guidance cell AO22X2 and target cell AO22X4 respectively. We apply the point cloud registration formulation to solve a geometrical transformation from the point cloud $C_g$ to the point cloud $C_t$. We obtain the geometrical transformation $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ by minimizing the least square sum of distances of any possible alignment of point cloud $C_g$ and point cloud $C_t$:

$$\min_{\mathcal{A}=\{\langle p_i^g, p_j^t \rangle\}} \min_{\mathbf{R},\mathbf{t}} \sum_{\langle p_i^g, p_j^t \rangle \in \mathcal{A}} \|(\mathbf{R}p_i^g + \mathbf{t}) - p_j^t\|^2 \qquad (1)$$

where $p_i^g$ and $p_j^t$ stand for the position vectors for the $i^{th}$ pin of the guidance cell, and the position vector for the $j^{th}$ pin of the target cell, respectively. $\mathcal{A} = \{\langle p_i^g, p_j^t \rangle\}$ represents the set of possible pin-to-pin alignments between point cloud $C_g$ and point cloud $C_t$. $f(p) = \mathbf{R}p + \mathbf{t}$ is the geometrical transformation. We apply the iterative closest point algorithm to get the optimized $\mathbf{R}$, $\mathbf{t}$ and $\mathcal{A}$ from Equation 1 [25]. The algorithm iteratively finds an alignment $\mathcal{A}$ and minimizes the least square difference under every alignment $\mathcal{A}$.

*2) Steiner tree generation:* To further migrate the whole routing topology, our framework constructs a rough Steiner tree for each net of the target cell. A Steiner tree contains the Steiner points and edges representing their connection information. We obtain the pin-to-pin alignment $\mathcal{A} = \{\langle p_i^g, p_j^t \rangle\}$ between the guidance net and the target net from the aforementioned geometrical transformation. We transfer the edges from the Steiner tree of the guidance net in

[1]Note that we study the routing problem on multiple metal layers (normally on M1 and M2). To make the algorithm process easy to understand, here we purposely choose a case with routing paths merely on M1.

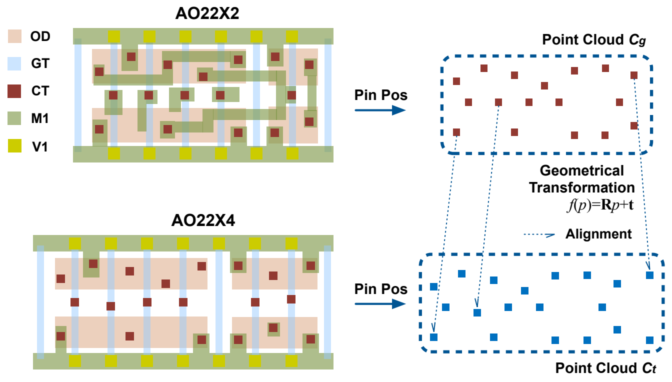

Fig. 3: An example of geometrical transformation: the geometrical transformation between the pins of AO22X2 and the pins of AO22X4.[1]

a Breath-First-Search (BFS) manner. Pins of the target net act as leaf-level Steiner points of the Steiner tree. The BFS traverses the whole tree from the leaf-level points. If the two points of a visited edge have corresponding points in the target net, then the algorithm adds an edge between the two corresponding points to the Steiner tree. The algorithm starts from the pins and propagates to all the connected Steiner points. For pins not connected to the Steiner tree, the algorithm adds an edge between the pin with the closest Steiner point and completes its routing by subsequent constrained A-star routing.

## *B. Planning for Design Rules*

In order to eliminate design rule violations and improve the routability for all nets, the algorithm further refines the positions of the Steiner points and the selection of vias. As described in Section III-A, the generation process of the Steiner points and vias focuses on a local net each time. We introduce an approach to consider how the Steiner points affect the routing resources of the whole layout.

The key idea of our approach is to reduce the conflicts over the repetitive requests of the same routing resources. For example, different nets going through an overlapping area on the M1 layer can cause such conflicts. Besides, spacing rules set out the minimum spacing requirements for two segments on the same layer, and area rules set out the minimum area requirements for a connected shape on a layer, which introduces more variants of the conflicts. To resolve the resource conflicts, we estimate the usage of routing resources by balancing the wire *density*.

We define the *density* as the ratio of the overlapped area of Steiner tree segment. As shown in Figure 4, the fine-grained grid is the basic routing grid and we split the routing resources into coarse-grained grids. We define *density* as the usage rate of each coarse-grained grid. To be more specific, we compute *density* for a grid as:

$$\begin{aligned} density &= \sum_{w=(x_l^w, y_l^w, x_h^w, y_h^w)} x_{intersect} * y_{intersect} / A_{grid} \\ x_{intersect} &= \max(0, \min(x_h^w, x_h^v) - \max(x_l^w, x_l^v)) \\ y_{intersect} &= \max(0, \min(y_h^w, y_h^v) - \max(y_l^w, y_l^v)) \end{aligned} \qquad (2)$$

where $w$ is an arbitrary net segment, i.e., an edge or a via point from a Steiner tree. And $x_l^w, y_l^w, x_h^w, y_h^w$ represents the coordinates of bottom-left and top-right corners for net segment $w$, and

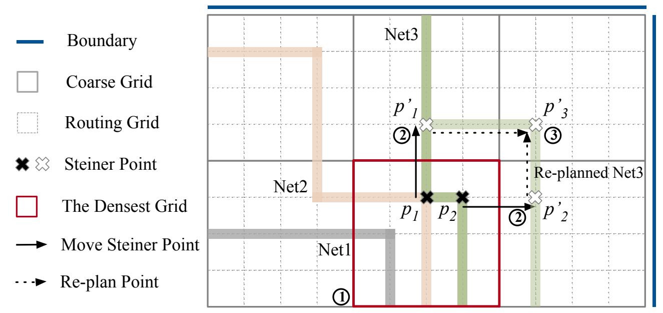

Fig. 4: Routing planning: ① Find grid with the highest density; ② Move Steiner points $p_1, p_2$ of Net3 to adjacent grids with new Steiner points $p_1', p_2'$; ③ Re-plan Steiner points with added Steiner point $p_3'$.

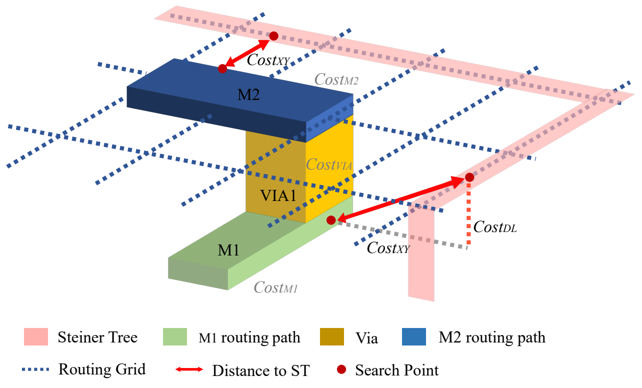

Fig. 5: Part of the routing cost types: unit wire cost, via cost, and 3D guidance cost.

$x_l^v, y_l^v, x_h^v, y_h^v$ for the grid. For area rules, we determine an additional expansion to a net segment based on whether the segment is separate from other segments on the same layer and has insufficient area. By computing the *density* for each grid, we construct a density map of all grids.

We employ an iterative process to sweep the net segments from the densest grid to adjacent grids. Each iteration of the process consists of three steps to re-plan the generated Steiner trees. Figure 4 describes the three steps of the iterative process. The first step is to find the densest grid in the calculated density map. Inside the densest grid, we check the spacing rules between different nets and identify the conflicts of routing resources. As shown in Figure 4, the Steiner tree of Net2 and Net3 appear in the densest grid and request the same routing resources. We specify that the preferred direction for sweeping is towards the boundary of the layout. Net3 is closer to the boundary and therefore the algorithm sweeps the segments of Net3 to adjacent grids. The second step is to move the Steiner points of Net3 to the adjacent grid to resolve the routing resource conflict. The algorithm finds the nearest routing grid in adjacent grids for the Steiner points $p_1$ and $p_2$ of Net3. The routing grids become the new Steiner points $p_1'$ and $p_2'$. The third step is to re-plan those Steiner points in order to restore the tree topology. We start from $p_1'$ and $p_2'$ and draw lines parallel to the original segments of the Steiner tree. The point where lines intersect outside the densest grid becomes the new Steiner point $p_3'$. After re-planning the Steiner points, we traverse all net segments to remove redundant Steiner point like $p_2'$. The algorithm repeats the above steps until the density of every grid is less than a threshold. As the original Steiner trees imitate the routing topology of manual guidance, the algorithm usually converges to a reasonable density map in a few iterations. The algorithm provides well-planned Steiner trees of each net for the next routing stage.

### C. Constrained A-star Routing

After generating well-planned Steiner trees, our framework completes routing paths with a pathfinder algorithm on more fine-grained routing grids. We constrain the A-star algorithm to find routing paths that follow the trajectory of the Steiner trees. A conventional A-star flow gives the routing solution to one net by connecting the pins one by one. At one step of the A-star flow, for a pin that has not been connected to the corresponding net, the algorithm finds a path with the smallest cost connecting the pin and the routed part of the net. The algorithm is concerned with two perspectives: in which order we connect the pins of a net, and how we decide the cost of a routing path. We explain how we adapt the A-star algorithm from those two perspectives:

*1) Pin order by MST:* We give a pin order for routing each net by a minimum spanning tree (MST). We construct a clique graph by regarding pin as node connecting all nodes. The edge cost between two nodes is the path length between the corresponding pins. The path length is defined on the Steiner tree, i.e., the sum of segment length between two pins. Solving MST by the Kruskal algorithm gives the order to route the pins.

*2) 3D guidance cost:* The A-star algorithm is basically guided by the routing costs. The routing costs usually include the unit wire cost, the via cost, and the bending cost. Also, the algorithm avoids searching for those points that violate the design rules. In addition, we add a 3D guidance cost to our search policy.

Figure 5 illustrates several types of routing costs. To route as close as possible to the Steiner tree generated from the previous stage, the algorithm estimates a 3D guidance cost that reflects the distance from where the search point is supposed to be. For the search point on the M2 routing path shown in Figure 5, the 3D guidance cost is the distance $Cost_{XY}$ from the point to the closest Steiner tree edge. For another one on the M1 routing path, as all Steiner tree edges close to the point are on the M2 layer, aside from the projected distance $cost_{XY}$, the 3D guidance cost contains the penalty $cost_{DL}$ for not being on the correct layer. Constrained by the 3D guidance cost, our framework imitates the routing pattern from the guidance layout and generates a well-behaved layout for the target cell.

## IV. Experimental Results

We conduct our experiments on SMIC28 (28nm) PDK. The SMIC28 PDK allows bending on both M1 and M2 metal layers. We implement our method using C++ on a virtual CentOS7 platform, with 8 cores Intel Xeon Gold 6230 CPU at 2.10GHz and 16GB memory. We perform DRC, LVS, and PEX with Mentor Calibre [26], and report the results of cell characterization with Primarius NanoCell [27]. We adopt manual standard cell designs and layouts from the real-world customized standard cell library of an industrial IP design team for experiments.

As for baselines, we compare with a customized SMT-based synthesis method and an adapted layout compaction method. The baseline methods share the same placement with our method. We adapt the SMT-based cell synthesis algorithm [11] to SMIC28. It should be noted that the SMT-based method cannot deal with area design rules, because it is hard to model the area of an isolated metal shape as an SMT formula. For the compaction method, we refer to [24] and perform the SMT-based method for those nets which have different numbers of pins compared to the guidance layout.

TABLE I: THE GEOMETRICAL RESULTS OF CASE STUDY I

| Cell | CELL INFO | | RUNTIME | | | M2 Usage | | | | #VIAs | | | | #DRVs | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #T | #NETs | SMT [11] | Compact [24] | Ours | Manual | SMT [11] | Compact [24] | Ours | Manual | SMT [11] | Compact [24] | Ours | Manual | SMT [11] | Compact [24] | Ours |
| AND | 6 | 9 | 1.75s | 1.34s | 0.37s | 0% | 12.2% | 12.3% | 0% | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| NAND | 4 | 8 | 0.84s | 1.37s | 1.88s | 0% | 13.5% | 10.0% | 0% | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| NOR | 4 | 8 | 0.82s | 1.56s | 1.62s | 0% | 12.9% | 13.1% | 0% | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| DEL | 8 | 9 | 1.37s | 1.67s | 0.46s | 0% | 19.5% | 8.7% | 0% | 0 | 4 | 2 | 0 | 0 | 2 | 0 | 0 |
| AO | 10 | 13 | 3.37s | 2.54s | 0.48s | 21.0% | 21.9% | 20.7% | 20.2% | 6 | 6 | 6 | 6 | 0 | 0 | 0 | 0 |
| Latch | 16 | 15 | 14.9s | 16.83s | 2.69s | 17.6% | 26.7% | 17.9% | 17.5% | 6 | 16 | 10 | 6 | 0 | 6 | 0 | 0 |
| D-FlipFlop | 24 | 18 | ¿1h | 301.3s | 20.45s | 22.2% | - | 27.1% | 23.5% | 16 | - | 18 | 16 | 0 | - | 0 | 0 |
| SD-FLipFlop | 32 | 26 | ¿1h | 578.2s | 17.8s | 23.0% | - | 32.8% | 24.1% | 20 | - | 24 | 20 | 0 | - | 0 | 0 |

* SMT, Compact and Ours share the same placement results, and hence their area metrics are the same.

TABLE II: THE CHARACTERIZATION RESULTS OF CASE STUDY I

| cell | Norm. Delay Rise | | | Norm. Delay Fall | | | Norm. Power Rise | | | Norm. Power Fall | | | Norm. Transition Time Rise | | | Norm. Transition Time Fall | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SMT % | Compact % | Ours % | SMT% | Compact % | Ours % | SMT % | Compact % | Ours % | SMT % | Compact % | Ours % | SMT % | Compact % | Ours % | SMT % | Compact % | Ours % |
| AND | 0.041% | 0.031% | **-0.112%** | 0.051% | 0.021% | **-0.078%** | 0.773% | 0.637% | 0.145% | 0.058% | 0.058% | **-0.128%** | 0.001% | -0.063% | **-0.117%** | -0.011% | 0.013% | **-0.017%** |
| NAND | 0.478% | 0.012% | **-2.435%** | 1.364% | 0.002% | **-2.640%** | 1.965% | 1.302% | **-2.070%** | 3.978% | 4.678% | 4.918% | -0.019% | -0.034% | **-2.607%** | 0.002% | -0.834% | **-2.619%** |
| NOR | 1.123% | 1.077% | **-1.827%** | 0.535% | 0.671% | **-1.922%** | 3.024% | 0.668% | **-0.638%** | **-0.039%** | 0.476% | 0.364% | 0.604% | -1.918% | **-2.201%** | -0.291% | 0.174% | **-1.800%** |
| DEL | 0.002% | 0.034% | **-0.076%** | 0.012% | **-0.998%** | -0.060% | 2.943% | 1.255% | **-0.107%** | 1.026% | 0.008% | **-0.195%** | -0.128% | 0.064% | **-0.128%** | 0.911% | **-0.024%** | -0.022% |
| AO | 0.250% | -1.354% | **-3.929%** | 3.947% | -3.018% | **-3.729%** | 7.659% | 4.887% | 4.852% | 1.071% | 1.282% | 1.371% | -0.002% | -0.316% | **-2.019%** | 1.277% | 1.012% | **-0.516%** |
| Latch | 0.071% | 1.433% | **-0.034%** | 1.085% | 0.077% | **-0.034%** | 1.691% | 0.058% | 0.010% | 1.734% | 1.655% | 0.008% | 1.003% | 0.954% | **-0.094%** | -0.058% | -0.036% | **-0.064%** |
| D-FlipFlop | - | 4.702% | 1.371% | - | 4.039% | 2.067% | - | 1.095% | 0.367% | - | 2.003% | 1.079% | - | 3.245% | 0.942% | - | 2.567% | 0.943% |
| SD-FLipFlop | - | 3.676% | 0.249% | - | 0.998% | 0.182% | - | 1.577% | **-0.036%** | - | 2.698% | **-0.011%** | - | 2.250% | 0.825% | - | 0.633% | 0.662% |

* Terms are shown as the ratio of the difference compared with Manual results, and hence Manual results are all 0%.
Bold indicates the best result in the three methods and is better than the manual result.

We conduct two case studies to validate our framework. The first case study is validated on a batch of cells by migration from drive strength X2 to X4. We adopt the widely used fan-out-of-4 metrics to show the characterization results of delay, power, and transition time [28], [29].

## A. Case Study I

This case study shows the results of migrating cells from drive strength X2 to X4. The geometrical results are shown in Table I. Standard cells with various numbers of transistors ("#T") and nets ("#NETs") are considered, from simple AND to complex SD-FlipFlop. We report runtime, M2 usage, number of vias, and number of DRC violations. The M2 usage and vias demonstrate the quality of routing. We calculate M2 usage as the ratio of M2 wirelength to the M1 wirelength used by the Manual layout. Our method can generate DRC-clean solutions on all cases with comparable usage of M2 layer and vias to manual layouts, while the baselines cannot guarantee DRC clean and tend to use more M2 tracks. As the compaction method cannot handle the changed number of pins, we use the SMT-based method for those nets which uses more M2 tracks and vias. For complex cells like SD-FlipFlop, our method can finish in about 20 seconds while the SMT-based method failed to find solutions and the compaction method takes a much longer time. These results demonstrate that our method migrates cells efficiently without DRC violations and achieves similar utilization of routing resources to manual layouts.

Table II depicts the normalized electrical performance of manual layouts and ours. We compare the delay, power, and transition time at rise and fall conditions. The terms listed in Table II are the ratio of the difference between the manual and the method, i.e., the smaller the better. Overall, our method can achieve competitive performance compared with manual layouts. In particular, 6 out of 8 cells achieve better delay rise, 4 out of 8 cells gain better power rise, 6 out of 8 cells achieve better transition time rise, and 5 out of 8 cells achieve better transition time fall. Our approach is not only geometrically similar to the manual layout but also achieves competitive electrical performance.

## B. Case Study II

Table III depicts the results of migrating BUF cells with various scaling sizes, i.e., X2 to X4, X4 to X8, X8 to X16, and X16 to X32.

TABLE III: THE RESULTS OF CASE STUDY II ON BUF CELL

| Exp. | Runtime | | M2 Usage | | | #VIAs | | | Delay (%) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SMT [11] | Ours | Manual | SMT [11] | Ours | Manual | SMT [11] | Ours | SMT [11] | Ours |
| X1→X2 | 0.58s | 0.22s | 0% | 0% | 0% | 0 | 0 | 0 | 0.967% | 0.911% |
| X2→X4 | 0.7s | 0.46s | 0% | 16.2% | 0% | 0 | 2 | 0 | 0.731% | -0.565% |
| X4→X8 | 1.1s | 0.57s | 0% | 23.9% | 0% | 0 | 4 | 0 | 1.665% | -0.340% |
| X8→X16 | 3.3s | 0.87s | 0% | 13.1% | 0% | 0 | 2 | 0 | 0.495% | -0.436% |
| X16→X32 | 36.9s | 0.91s | 0% | 12.4% | 0% | 0 | 2 | 0 | 0.994% | 0.219% |

* All results listed here are free from DRC violations.

We compare the geometrical metrics between manual layouts, SMT-based layouts, and ours, and further compare differences in delays between manual layouts and ours. The compaction-based method cannot synthesize a DRC-clean layout for the last two tasks (X8 to X16, X16 to X32), and therefore not listed in the table. Our method accelerates layout generation by 2 to 40 times compared to SMT-based method [11] and accomplishes all cases in less than one second. We can achieve zero utilization of M2 metal and vias, keeping the same routing quality as manual layouts, while the SMT-based method starts to use more M2 metal and vias as cells scale up. Case Study 2 demonstrates the generality and routing quality of our method when dealing with various scaling sizes.

## V. CONCLUSION

In this paper, we present a standard cell layout framework to migrate cells by *routing imitation*. We explore the geometrical homogeneity among cells of multiple drive strengths and migrate both the geometrical information and the structural information of a layout topology to abstract guidance for the target cell. Our framework then completes the layout synthesis under the generated guidance. Experimental results on an industrial standard cell library show that our framework can generate layouts with highly competitive performance to manual layouts in an extremely short time. Future work includes validating the algorithm to support cell libraries in advanced technology nodes.

## ACKNOWLEDGE

This project is supported in part by the National Natural Science Foundation of China (Grant No. 62141404, 62125401) and the 111 project (B18001).

## REFERENCES

[1] P. Van Cleeff, S. Hougardy *et al.*, "Bonncell: Automatic cell layout in the 7-nm era," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 10, pp. 2872–2885, 2019.

[2] B. Taylor and L. Pileggi, "Exact Combinatorial Optimization Methods for Physical Design of Regular Logic Bricks," in *ACM/IEEE Design Automation Conference (DAC)*, 2007, pp. 344–349.

[3] N. Ryzhenko and S. Burns, "Standard cell routing via Boolean satisfiability," in *ACM/IEEE Design Automation Conference (DAC)*, 2012, pp. 603–612.

[4] J. Cortadella, J. Petit *et al.*, "A Boolean Rule-Based Approach for Manufacturability-Aware Cell Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 3, pp. 409–422, 2014.

[5] A. Lu, H.-J. Lu *et al.*, "Simultaneous transistor pairing and placement for CMOS standard cells," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2015, pp. 1647–1652.

[6] P. Debacker, K. Han *et al.*, "Vertical M1 routing-aware detailed placement for congestion and wirelength reduction in sub-10nm nodes," in *ACM/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[7] K. Han, A. B. Kahng *et al.*, "Evaluation of BEOL design rule impacts using an optimal ILP-based detailed router," in *ACM/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[8] X. Xu, B. Cline *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 (TCAD)*, vol. 34, no. 5, pp. 699–712, 2015.

[9] W. Ye, B. Yu *et al.*, "Standard Cell Layout Regularity and Pin Access Optimization Considering Middle-of-Line," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, New York, NY, USA, 2015, p. 289–294.

[10] H.-J. Lu, E.-J. Jang *et al.*, "Practical ILP-based routing of standard cells," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2016, pp. 245–248.

[11] D. Lee, D. Park *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 (TCAD)*, vol. 40, no. 10, pp. 2142–2155, 2020.

[12] Y.-L. Li, S.-T. Lin *et al.*, "NCTUcell: A DDA-Aware Cell Library Generator for FinFET Structure with Implicitly Adjustable Grid Map," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[13] J. Seo, J. Jung *et al.*, "Pin accessibility-driven cell layout redesign and placement optimization," in *ACM/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[14] K. Jo, S. Ahn *et al.*, "Cohesive techniques for cell layout optimization supporting 2D metal-1 routing completion," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2018.

[15] X. Xu, B. Yu *et al.*, "PARR: Pin access planning and regular routing for self-aligned double patterning," in *ACM/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[16] W.-T. J. Chan, P.-H. Ho *et al.*, "Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning," in *ACM International Symposium on Physical Design (ISPD)*, ser. ISPD '17, New York, NY, USA, 2017, p. 15–21.

[17] A. C.-W. Liang, H.-M. Huang *et al.*, "Generating Layouts of Standard Cells by Implicit Learning on Design Rules for Advanced Processes," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2021, pp. 1829–1834.

[18] H. Ren and M. Fojtik, "Standard Cell Routing with Reinforcement Learning and Genetic Algorithm in Advanced Technology Nodes," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, New York, NY, USA, 2021, p. 684–689.

[19] H. Shin and C.-Y. Lo, "An Efficient Two-Dimensional Layout Compaction Algorithm," in *ACM/IEEE Design Automation Conference (DAC)*, 1989, pp. 290–295.

[20] J. Fang, J. Wong *et al.*, "A new fast constraint graph generation algorithm for VLSI layout compaction," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 1991, pp. 2858–2861 vol.5.

[21] D.-S. Fu, Y.-Z. Chaung *et al.*, "Topology-driven cell layout migration with collinear constraints," in *IEEE International Conference on Computer Design (ICCD)*, 2009, pp. 439–444.

[22] F. Fang and J. Zhu, "Automatic process migration of datapath hard IP libraries," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2004, pp. 888–893.

[23] J. Zhu, F. Fang *et al.*, "Calligrapher: a new layout-migration engine for hard intellectual property libraries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 24, no. 9, pp. 1347–1361, 2005.

[24] A. M. Ziesemer and R. A. d. L. Reis, "Simultaneous Two-Dimensional Cell Layout Compaction Using MILP with ASTRAN," in *2014 IEEE Computer Society Annual Symposium on VLSI*, 2014, pp. 350–355.

[25] K. S. Arun, T. S. Huang *et al.*, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[26] Mentor Graphics, "Calibre verification user's manual," 2020.

[27] "NanoCell," https://www.primarius-tech.com/en/products/NanoCell%20.

[28] M. Alioto, G. Palumbo *et al.*, "Understanding the effect of process variations on the delay of static and domino logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 5, pp. 697–710, 2010.

[29] M. Alioto, G. Scotti *et al.*, "A novel framework to estimate the path delay variability on the back of an envelope via the fan-out-of-4 metric," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 8, pp. 2073–2085, 2017.