Mixed-Cell-Height Legalization Considering Technology and Region Constraints[*]

Ziran Zhu[1], Xingquan Li[1], Yuhang Chen[1], Jianli Chen[1], Wenxing Zhu[1], and Yao-Wen Chang[2,3]
[1]Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350108, China
[2]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
[3]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
{N150320066, N130320024, N160320054, jlchen, wxzhu}@fzu.edu.cn; ywchang@ntu.edu.tw

ABSTRACT

Mixed-cell-height circuits have become popular in advanced technologies for better power, area, routability, and performance tradeoffs. With the technology and region constraints imposed by modern circuit designs, the mixed-cell-height legalization problem has become more challenging. In this paper, we present an effective and efficient legalization algorithm for mixed-cell-height circuit designs with technology and region constraints. We first present a fence region handling technique to unify the fence regions and the default ones. To obtain a desired cell assignment, we then propose a movement-aware cell reassignment method by iteratively reassigning cells in locally dense areas to their desired rows. After cell reassignment, a technology-aware legalization is presented to remove cell overlaps while satisfying the technology constraints. Finally, we propose a technology-aware refinement to further reduce the average and maximum cell movements without increasing the technology constraints violations. Compared with the champion of the 2017 ICCAD CAD Contest and the state-of-the-art work, experimental results show that our algorithm achieves the best average and maximum cell movements and significantly fewer technology constraint violations, in a comparable runtime.

1 INTRODUCTION

Mixed-cell-height circuits have become popular in advanced technologies for better power, area, routability, and performance tradeoffs [5]. Such multi-row-height standard cells incur great challenges for placement, especially the mixed-cell-height standard cell legalization, due to the heterogenous cell structures and additional power-rail constraints, as pointed out in [14]. Further, more complicated design rules and constraints need to be addressed, such as fence region constraints and technology constraints (e.g., cell edge spacing, pin shorts, and pin access issues). A legalizer without considering these constraints may produce illegal solutions or cause severe routing violations, which may be very difficult or time-consuming to correct these violations at the end of design flow [6]. Hence, it is desirable to consider mixed-cell-height legalization with these constraints.

To preserve solution quality from global placement or timing optimization as much as possible, an ideal legalization method should remove all cell overlaps and satisfy all design rules, while minimizing both the average and maximum cell movements [5]. In addition, the method should be fast and robust to handle a large number of cells in the state-of-the-art designs [5].

Recently, the mixed-cell-height standard cell legalization problem has been studied extensively [3, 4, 11, 14, 16]. The work [16] first proposed a detailed placement approach to handle designs with both of the single- and double-row height cells. In [4], a mixed-cell-height local legalization algorithm was proposed to place cells in a local region. Wang et al. [14] extended Abacus to handle the legalization problem with mixed-cell-height standard cells. The work [11] applied the legalizer in [4] to minimize the cell movement from global placement and then developed a density aware detailed placer for mixed-cell-height standard cell designs. Chen et al. [3] converted the mixed-cell-height legalization problem into a linear complementarity problem, and then solved the problem by a modulus-based matrix splitting iteration method (MMSIM). Unfortunately, none of these legalization methods considered the technology and fence region constraints of advanced node technologies.

On the other hand, existing routability driven legalizers considered the technology and fence region constraints for single-row height standard cell legalization problem in various ways. Darav et al. [6] and Huang et al. [8] extended Abacus to address the complicated design rules brought forward by modern technologies. In [9], a conservative legalization approach was proposed to prevent cells from being placed below routing blockages, and thus avoid detailed routing violations. Wang et al. [15] applied Abacus to legalize a placement, and then proposed a detailed placement algorithm to reduce cell edge spacing violations. Huang et al. [7] presented a detailed routability aware whitespace allocation technique during legalization to minimize pin short violations. However, none of these methods consider multi-row-height standard cells. In mixed-cell-height circuit designs, shifting a cell in one row may cause cell overlaps in another row. With a larger solution space and additional power-rail constraints, it is not easy to extend these works on single-row height standard cell placement with the technology and fence region constraints to handle mixed-cell-height designs effectively.

In order to encourage a legalizer to address the mixed-cell-height standard cell designs with various design rules in advanced node

[*]This work was partially supported by the National Natural Science Foundation of China under Grants 11501115, 61672005 and 11331003, and by AnaGlobe, MediaTek, TSMC, MOST of Taiwan under Grant No's MOST 104-2221-E-002-132-MY3, MOST 105-2221-E-002-190-MY3, MOST 106-2911-I-002-511, and NTU under Grant No. NTU-ERP-104R8951.

ICCAD '18, November 5–8, 2018, San Diego, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3240826

technologies, the 2017 ICCAD held a mixed-cell-height standard cell legalization contest [5]. The recent work [10] considered the contest problem and proposed a window-based cell insertion technique to legalize a placement, and developed two post-processing network-flow-based optimizations to further reduce the cell movement. Since the method legalizes one cell at a time in a window and tends to be local, the solution quality may be limited.

In this paper, we present an effective legalization algorithm to solve the mixed-cell-height designs with technology and fence region constraints. The major contributions of our work are summarized as follows:

- A fence region handling technique is presented to unify the fence regions and the default ones.
- Since a small number of disruptive cells are likely to cause excessive movement of a large number of cells, we propose a movement-aware cell reassignment method by iteratively reassigning disruptive cells in locally dense areas to their desired rows.
- We formulate the placement problem which considers the total and maximum movements and the technology constraints as a quadratic programming problem, and solve the problem by the modulus-based matrix splitting iteration method efficiently. Then, a minimum cost maximum flow algorithm is adopted to reassign the disruptive cells to rows.
- A technology-aware legalization is presented to remove cell overlaps and align each cell to a placement site, while satisfying the technology constraints.
- A technology-aware refinement is proposed to further reduce the average and maximum cell movements without increasing the technology constraints violations.
- Experimental results show that our algorithm is effective and efficient. Compared with the champion of the 2017 ICCAD CAD Contest [5] and the state-of-the-art work [10], our algorithm achieves the best average and maximum cell movements and significantly fewer technology constraint violations in comparable runtime.

The rest of this paper is organized as follows. Section 2 first introduces the design rules and constraints imposed by advanced node technologies, and then gives the problem statement. Section 3 details our legalization algorithm. Experimental results are shown in Section 4. Finally, conclusions are given in Section 5.

2 PRELIMINARIES

In this section, we first introduce the design rules and constraints in mixed-cell-height standard cell legalization, and then provide the problem statement.

2.1 Placement Constraints

The design rules and constraints of a placement include the power/ground (vdd/vss) alignment constraints and fence region constraints, as well as the technology constraints (e.g., edge spacing, pin access and pin shorts). We detail these constraints as follows.

2.1.1 *Power/Ground Alignment Constraints.*

Vdd/vss alignment constraints are due to the popularity of multi-row height cells. In modern design, the vdd and vss meshes are interlaced at the top and bottom of rows. Each cell has two types of pins named vdd and vss. Vdd/vss alignment constraints require that the vdd/vss pins of each cell must be aligned with vdd/vss meshes correctly.

For an odd-row-height cell, due to its different types of pins at the top and bottom, it can be placed to any desired rows directly or by vertical cell flipping to meet the alignment constraints. However, for an even-row-height cell, it must be placed to dedicated rows with proper vdd/vss meshes, since it has the same type of pins at the top and bottom, thus cell flipping cannot correct the vdd/vss mismatch.

Fig. 1 gives a placement example of four cells considering vdd/vss alignment constraints, in which the four cells c_1, c_2, c_3 and c_4 have vss pins at the bottom. As shown in the figure, for the odd-row-height cells c_1 and c_3, they not only can be placed to rows with vss mesh at the bottom directly, but also can be placed to rows with vdd mesh at the bottom by vertical cell flipping. However, for the even-row-height cells c_2 and c_4, they can only be placed to rows with vss mesh at the bottom, since they have the same type of pins at the top and bottom.

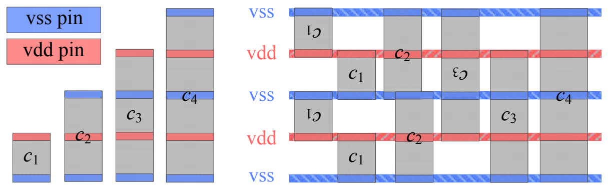


Figure 1: Example of vdd/vss alignment constraints on mixed-cell-height placement.

2.1.2 *Fence Region Constraints.*

In order to improve important design characteristics, sometimes cells with the same function need to be placed contiguously in a confined subregion [2]. Fence region constraints are imposed to handle issues such as reserving whitespaces for some specific cells and isolating separate voltage regions [2][6]. Each fence region can only be occupied by the cells assigned to the region.

A fence region can be non-rectangular, and is usually made up of multiple spatially joint or disjoint rectangular subregions. Fig. 2(a) shows the layout of des_perf_b_md1 with 12 fence regions in different colors [5], where fence regions have various shapes, and most of them are non-rectangular. Fig. 2(b) gives the layout of edit_dist_a_md2 with 1 fence region [5], which consists of 6 disjoint subregions. Such various shapes of fence regions pose a great challenge to legalization, and thus it is important to develop algorithms which can effectively handle the fence region constraints.

2.1.3 *Technology Constraints.*

To achieve better detailed routability, it is desirable to consider the technology constraints (e.g., pin short, pin access and edge spacing) in the legalization stage. It is more challenging to meet the technology constraints for the designs with mixed-cell-height cells, comparing with the designs with single-row height cells only. With multi-row height cells, shifting a cell to satisfy technology constraints in one row may violate the constraints in other rows.

The same as ICCAD 2017 CAD Contest [5], we consider those pin shorts/access issues that may be induced by fixed objects (including pre-routed wires and primary inputs/outputs). Pin shorts

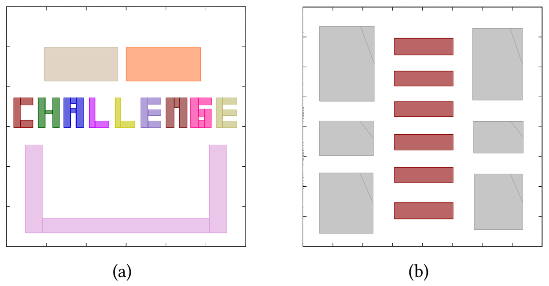


Figure 2: Example of fence region constraints. (a) The layout of des_perf_b_md1 with 12 fence regions in different colors [5]. (b) The layout of edit_dist_a_md2 with 1 fence region [5], which consists of 6 disjoint subregions.

constraints require that there are no overlap between cell pins (not including vdd/vss pins) and fixed objects on the same metal layer. Pin access issues are likely to arise when cell pins (not including vdd/vss pins) are placed below fixed objects. In addition, edge spacing constraints require a gap between two adjacent cells to prevent the occurrence of pin shorts.

Fig. 3 shows three examples of technology constraints. As shown in Fig. 3(a), the M2 (metal 2) pin overlaps with the M2 pre-routed mesh, resulting in the problem of pin short. In addition, the M1 (metal 1) pin is placed below the M2 pre-routed mesh, which may cause pin access issue, since the M1 pin may be blocked by the M2 mesh. Furthermore, Fig. 3(b) gives three possible placement scenarios that do not cause pin short/access issues. Fig. 3(c) shows an example of edge spacing constraints, where cell c_3 should meet three edge spacing constraints between c_3 and c_1, c_3 and c_2, and between c_3 and c_4.

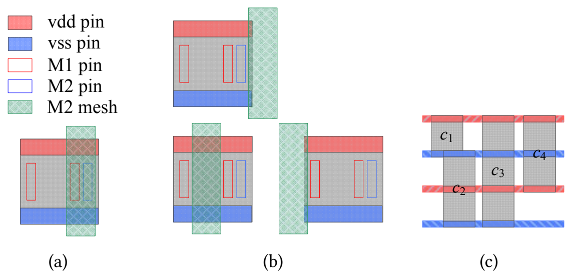


Figure 3: Examples of technology constraints. (a) Pin short issue occurs at which M2 pin overlaps with M2 mesh, and pin access issue occurs at which M1 pin is placed below M2 mesh and the pin may be blocked. (b) Three different placement scenarios that do not cause pin short/access issues. (c) Cell c_3 should meet three edge spacing constraints between c_3 and c_1, c_3 and c_2, and c_3 and c_4.

2.2 Problem Statement

A modern legalizer should be able to remove all cell overlaps, meet complicated design rules and constraints, and preserve the "good" solution provided by global placement as much as possible [5]. Given a global placement result of n mixed-cell-height standard cells $C = \{c_1, c_2, ..., c_n\}$ with technology and fence region constraints, where the width, height, the initial bottom-left corner positions of cell c_i and the placement row height are represented as w_i, h_i, (x'_i, y'_i), and R_h, respectively. The mixed-cell-height standard cell legalization considering technology and fence region constraints aims at determining the best position (x_i, y_i) of each cell c_i, such that the quality of the given placement result is preserved and the technology constraints are satisfied as much as possible. Therefore, our legalizer consider the average and maximum cell movements and violations of soft constraints (i.e., technology constraints), while the following hard constraints are met:

(1) cells must be non-overlapping,
(2) cells must be placed inside the chip region,
(3) cells must be located at placement sites on rows,
(4) cells must meet the fence region constraints, and
(5) cells must meet the power/ground alignment constraints.

According to the 2017 ICCAD CAD Contest [5], the hard constraints must be satisfied, and the fewer the number of violations of the soft constraints, the better the legalization result.

3 OUR ALGORITHM

Fig. 4 summarizes the overall flow of our algorithm, which mainly consists of four steps: (1) fence region handling, (2) movement-aware cell reassignment, (3) technology-aware legalization, and (4) technology-aware refinement. We shall detail these four major parts in the following subsections.

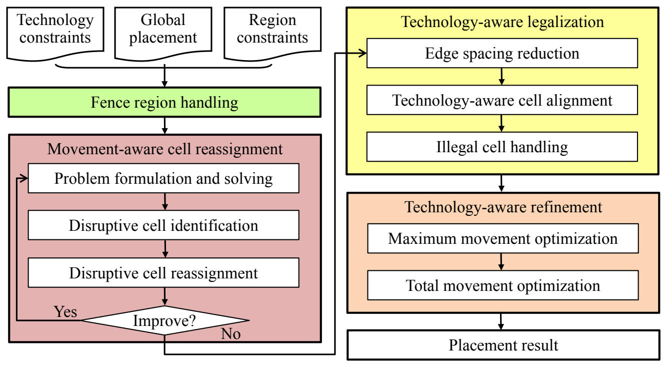

Technology constraints
Global placement
Region constraints
Fence region handling
Movement-aware cell reassignment
Problem formulation and solving
Disruptive cell identification
Disruptive cell reassignment
Yes
Improve?
No
Technology-aware legalization
Edge spacing reduction
Technology-aware cell alignment
Illegal cell handling
Technology-aware refinement
Maximum movement optimization
Total movement optimization
Placement result

Figure 4: Our algorithm flow.

3.1 Fence Region Handling

Since a fence region may not be rectangular and consists of multiple spatially joint or disjoint rectangular subregions, we first present a fence region handling technique to unify the fence regions and the default region (which is the region outside the given fence regions). More specifically, we decompose the design into several subchips, where each subchip corresponds to one region (a fence region or default region) and treats other regions as placement blockages. That is, each placement subchip has its own cells and placement blockages. After that, we can handle the legalization problem on each placement subchip (a fence region or default region) independently.

Fig. 5 shows two subchips, a default region and a fence region, decomposed from the design in Fig. 2(b), where blockages are in blue, fixed macros are in gray and blank spaces are in yellow. Fig. 5(a) shows the default region which the fence region is considered as blockages, and Fig. 5(b) shows the fence region which the default region is considered as a blockage. Hence, a cell that is assigned to the exclusive region must be placed within this fence region; otherwise, cells must be placed in the default region.

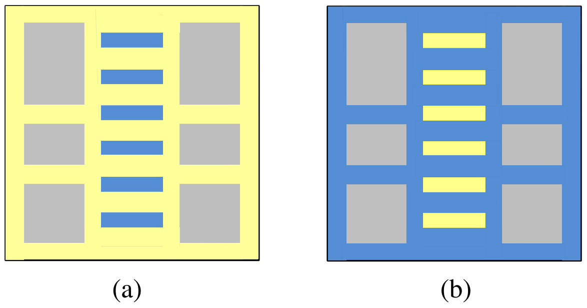


Figure 5: Design region in Fig. 2(b) is decomposed into two subchips (a default region and a fence region), where blockages are in blue, fixed macros are in gray and blank spaces are in yellow. (a) In the default region legalization, each fence subregion is considered as a blockage. (b) In the fence region legalization, the default region is considered as a blockage.

For each cell, our algorithm first aligns it to its corresponding subchip and its nearest correct rows. The word "correct" means that (1) cells should be placed inside the corresponding subchips and cannot overlap with a fixed macro or a blockage, which naturally satisfies the fence region constraints; (2) cells should meet the power/ground alignment constraints; and (3) any signal pin (excluding vdd/vss pin) on metal layer k should not overlap with the horizontal pre-routed wires on metal layer k or $k+1$, so that the pin short/access issues would not be caused by the horizontal pre-routed wires. After that, the positions of each cell c_i is updated to (x_i^0, y_i^0) from (x_i', y_i').

3.2 Movement-Aware Cell Reassignment

After aligning each cell to its corresponding subchip and its nearest correct rows, it is obvious that the total cell movement in the vertical direction is minimized. In addition, if the cell distribution in a row is locally sparse, then the overlaps among these cells in the row can be solved desirably; in contrast, if the cell distribution in a row is locally dense, then resolving the overlaps among these cells in the row might lead to excessive cell movement, and the locations of some cells in the row may be undesired. In this subsection, we present a movement-aware cell reassignment method by iteratively reassigning some cells in locally dense areas to their desired rows.

Our movement-aware cell reassignment is iteratively operated as follows. At each iteration, based on the current cell assignment and cell ordering, relaxing the right boundary constraints, we first formulate the legalization problem which considers the total and maximum movements and the technology constraints as a quadratic programming problem, and solve the problem by the modulus-based matrix splitting iteration method (MMSIM) [3] optimally to obtain a placement result. Then, based on the obtained placement result, we identify disruptive cells that are likely to be responsible for an excessive movement. Furthermore, we perform a minimum cost maximum flow algorithm [13] to reassign the disruptive cells to their best possible rows. The process stops if the score function (4) defined in Section 4 is not further improved. Finally, we obtain a desired cell reassignment result.

3.2.1 Problem Formulation and Solving. Let $C_r \subseteq C$ be the set of cells assigned to a placement subchip r. Based on the current cell assignment and cell ordering, and relaxing the right boundary constraints, similar to [3], the mixed-cell-height legalization problem for each placement subchip r can be formulated as a quadratic program as follows:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\sum_{c_i \in C_r} (y_i^0 - y_i')^2 (x_i - x_i')^2 \\ \text{s.t.} \quad & x_j - x_i \geq w_i + e_{i,j},\ \forall c_i, c_j \in C_r,\ \text{if } x_j^0 \geq x_i^0 \\ & \text{and } c_i \text{ and } c_j \text{ are adjacent in the same row;} \\ & x_i \geq 0,\ \forall c_i \in C_r. \end{aligned} \tag{1}$$

where (x_i', y_i') is the initial position of cell c_i, (x_i^0, y_i^0) is the position of cell c_i after aligning cells to their nearest correct rows, and $e_{i,j}$ is the required minimum spacing between the right edge of cell c_i and left edge of cell c_j.

The objective function of Problem (1) is the weighted sum of the squared cell movement, where $(y_i^0 - y_i')^2$ is the weight of the horizontal movement of cell c_i. If the vertical movement of cell c_i is larger than the vertical movement of cell c_j, i.e., $(y_i^0 - y_i')^2 > (y_j^0 - y_j')^2$, then the objective function would tend to make the horizontal movement of cell c_i smaller than the horizontal movement of c_j, thus preventing the maximum cell movement from being too large. In addition, the constraints of Problem (1) assure that there is no overlap among the cells and all the edge spacing constraints are satisfied.

Since we need to iteratively solve Problem (1), it requires that the solution method is effective and efficient. In [1], Bai proposed a linear-time MMSIM for solving the linear complementarity problem. Further, Chen et al. [3] adopted the MMSIM to solve the mixed-cell-height legalization problem under the premise of fixing the cell order and relaxing the right boundary constraints. In this paper, we also adopt the MMSIM to solve Problem (1).

Note that there are two minor differences between Problem (1) and the original formulation in the work [3]. That is, we add the vertical movement of each cell as a weight to the objective function, and the lower bound of each inequality constraint is replaced by $w_i + e_{i,j}$ from w_i to consider the edge spacing. Benefited from these two differences, we can minimize the total and maximum cell movements as much as possible, while removing cell overlaps and satisfying the edge spacing constraints.

In order to satisfy the convergence condition of the MMSIM solver in [3], we introduce multiple variables to represent a multi-row height cell. More specifically, if c_i is a single-row-height cell, we introduce the variable x_{i1} for this cell; otherwise, we introduce the variables $x_{i1}, x_{i2}, \ldots, x_{id}$ for a multi-row height cell c_i, where d is equal to cell height h_i divided by the row height, i.e., $d = \frac{h_i}{R_h}$. Then, similar to the work [3], we can rewrite Problem (1) as

$$\begin{aligned} \min \quad & \tfrac{1}{2}x^T Q x + p^T x \\ \text{s.t.} \quad & Bx \geq b, \\ & Ex = 0, \\ & x \geq 0, \end{aligned} \tag{2}$$

Algorithm 1 Disruptive Cells Identification

```
Require: A set $C_r$ of cells in the placement subchip $r$.
Ensure: Remove the disruptive cells from $C_r$.
1: Distribute $C_r$ into groups;
2: for each group $g$ do
3:     Initialize $g_w$, $g_{xmin}$, and $g_{xmax}$;
4:     while $g_w - (g_{xmax} - g_{xmin}) > R_h$ do
5:         Calculate the cost of each cell in the group;
6:         Mark the cell with the highest cost and remove it from $C_r$
           and the group;
7:         Update $g_w$.
8:     end while
9: end for
```

where Q is a diagonal matrix with its diagonal elements $q_{i,i} = (y_i^0 - y_i')^2$, p is a vector with the i-th component $p_i = -(y_i^0 - y_i')^2 x_i'$, B is a constraint matrix with only two nonzero elements -1 and 1 in each row, in which the number of rows gives the number of constraints, and the number of columns equals that of variables. The constraint $Ex = 0$ ensures that the variables for each multi-row height cell are equal. Obviously, Q is a symmetric positive definite matrix, and B is of full row rank, and thus the convergence of the MMSIM solver in [3] can be guaranteed.

Finally, we use the MMSIM solver in [3] to solve Problem (1), which only needs linear time for each iteration. In addition, according to Theorem 2 of [3], we also have the following theorem:

Theorem 1. *The solution generated by the MMSIM solver gives the optimal solution of Problem (1).*

3.2.2 Disruptive Cell Identification. Based on the placement result obtained by Problem (1), in order to obtain a desired cell assignment, it is crucial to find out the disruptive cells that are likely to be responsible for an excessive horizontal movement, and reassign these cells to other rows. We first use a sliding window to determine which local area has a larger horizontal movement. If the average movement of cells in the window is greater than 2× the average movement of cells in the subchip, then we put the cells in the window back to their original x-coordinates.

Algorithm 1 gives an overview of the disruptive cell identification. In Line 1, we first distribute all cells in each row into groups, and there is no space between any two adjacent cells in a group. More specifically, for any two adjacent cells c_i and c_j in the same group, if $x_i^0 \le x_j^0$, then it must satisfy $x_i^0 + w_i \ge x_j^0$. Then, for each group g, we initialize g_w, g_{xmin} and g_{xmax} in Line 3, where g_w, g_{xmin}, and g_{xmax} represent the total width of cells, the minimum left x-coordinate of cells and the maximum right x-coordinate of cells in the group, respectively. If $g_w - (g_{xmax} - g_{xmin}) > R_h$, then we regard the group g as a locally dense group, which probably contains some cells that are unreasonably assigned and need to be reassigned.

In Line 5, we calculate the cost of each cell in the locally dense group. The cost function for each cell c_i is defined as

$$cost_i = c_i^o + \beta_1 c_i^d + \beta_2 c_i^v, \tag{3}$$

where β_1 and β_2 are user-defined parameters, c_i^o is the total overlap between cell c_i and other cells in the same row, $c_i^d = \frac{g_{xmax} - g_{xmin}}{2} - |x_i^0 + \frac{w_i}{2} - \frac{g_{xmax} + g_{xmin}}{2}|$, and c_i^v is set as $|y_i^0 - y_i'|$ if the initial position of cell c_i is within the subchip and does not overlap with any fixed macro or blockage; otherwise, c_i^v is set as 0.

In Line 6, we mark a cell with the highest cost as a disruptive cell and remove it from C_r and the group temporarily. Finally, we update w_g in Line 7 and repeatedly mark a cell with the highest cost as a disruptive cell until $g_w - (g_{xmax} - g_{xmin}) \le R_h$.

Algorithm 2 Disruptive Cells Reassignment

```
Require: Positions of the cells in $C_r$ except disruptive cells.
Ensure: Positions of disruptive cells.
1: for each cell type $T$ do
2:     Initialize $C_{DT} = \emptyset$, $S_{DT} = \emptyset$, $f = 1$, $f_{min}$;
3:     for each disruptive cell $c_i \in C_T$ do
4:         $C_{DT}.push(c_i)$;
5:         Initialize $TS = \emptyset$, $mm$;
6:         while $|TS| < 5$ do
7:             $TS \leftarrow$ collect blank spaces which can contain the
               rectangle$(h_i, w_i \times f)$ within $mm$ of initial positions of
               cell $c_i$;
8:             $f \leftarrow max(f_{min}, f - 0.1)$;
9:             $mm \leftarrow mm + R_h$;
10:        end while
11:        $S_{DT} \leftarrow S_{DT} \cup TS$;
12:    end for
13:    $SS_{DT} \leftarrow$ divide each space in $S_{DT}$;
14:    A flow network $G = (C_{DT}, SS_{DT}, C_{DT} \times SS_{DT})$ is con-
       structed, and the positions of disruptive cells are found.
15: end for
```

3.2.3 Disruptive Cell Reassignment. Algorithm 2 shows an overview of the disruptive cell reassignment. For each cell type of the disruptive cells, we would first search some possible blank spaces (Lines 2-13), and then construct a flow network of these cells and blank spaces. Finally, a minimum cost maximum flow algorithm [13] is adopted to find the best location for each disruptive cell in Line 14.

In order to prevent excessive cell movements, especially the maximum cell movement, we allow that the width of a blank space to be smaller than the width of a cell, and define a tolerable maximum movement threshold mm as well as a tolerable width threshold f_{min} when searching the blank spaces for disruptive cells. In Line 2 of Algorithm 2, C_{DT} is the set of disruptive cells with the cell type T, S_{DT} is the set of blank spaces for the cells in C_{DT}, and f is the width factor. In Lines 3-12, for each disruptive cell c_i, we find at least five blank spaces which can contain the rectangle $(h_i, w_i \times f)$ with the tolerable maximum movement threshold mm of the initial position of c_i. If we cannot find enough blank spaces, then we decrease f and increase mm and find blank spaces again.

Since a blank space can accommodate multiple cells, each blank space is divided into a set of sub-blank spaces in Line 13, and each sub-blank space can only accommodate one cell. Then, a flow network is constructed in Line 14, where the capacity of all edge is one, and the cost of assigning the cell $c_i \in C_{DT}$ to the sub-blank space $ss \in SS_{DT}$ is defined as

$$cost_i^{ss} = m_i^{ss} + \max\{0, \frac{n_{ss}(w_i - w_{ss})}{2}\},$$

where m_i^{ss} is the Manhattan distance between cell c_i and sub-blank space ss, n_{ss} is the number of cells abutting the left border and the right border of the sub-blank space. Finally, the minimum cost maximum flow algorithm [13] is applied to obtain a location for each disruptive cell.

3.3 Technology-Aware Legalization

After the movement-aware cell reassignment, there may still be possible cell overlaps due to the precision of computation. Besides, some cells may be out of the right boundary since we relax the right boundary constraints in the Problem (2). In this subsection, we remove all cell overlaps and align cells to the placement sites, while completely eliminating the edge spacing violations.

3.3.1 Edge Spacing Reduction. Since the types of the left and right edges of a cell may be different, we can reduce the edge spacing issues by flipping some cells horizontally. Hence, after obtaining a better cell assignment in Subsection 3.2, we take a simple, yet effective operation to reduce the cell edge spacing issues. That is, we sort cells by their left x-coordinates in non-decreasing order, and check all cells one by one to see if the edge spacing issues can be reduced by flipping the cells horizontally. Once a cell has been checked, it will not be flipped horizontally again. For each cell c_i, let $N(i)$ be the set of cells adjacent to c_i in rows. We flip the cell c_i horizontally if and only if the following two conditions are satisfied: (1) the required minimum spacing between any cell in $N(i)$ and cell c_i does not increase, (2) there exists $c_j \in N(i)$ such that the required minimum spacing between c_i and c_j is decreased.

3.3.2 Technology-Aware Cell Alignment. Due to the movement-aware cell reassignment in Subsection 3.2, cell overlaps may occur. We remove all these cell overlaps with the minimum cell movement, while completely eliminating the edge spacing violations from a global view by modifying the MMSIM solver [3] to solve Problem (2) again.

After solving Problem (2) by the MMSIM solver, we need to further align cells to the placement sites and fix the illegal cells that overlap with fixed macros or are out of the right boundary, and address the pin short and pin access issues. The pin short and pin access violations occur when the signal pins (excluding the vdd/vss pins) of cells overlap with fixed objects on the same layer (named pin short) or the upper layer (named pin access). The fixed objects include horizontal pre-routed wires, vertical pre-routed wires, and primary inputs/outputs. Since we have ensured that any signal pin (excluding vdd/vss pin) on metal layer k cannot overlap with the horizontal pre-routed wires on metal layer k or $k+1$ during the cell assignment, we then only need to eliminate the pin short and pin access violations caused by vertical pre-routed wires and primary inputs/outputs.

In order to legalize the placement and reduce the technology violations, while preserving the optimal solution from the MMSIM solver as much as possible, we first define an allowed maximum movement *amm* for cells based on the current cell positions. Then, we sort cells by their left x-coordinates in non-decreasing order, and place the cells on placement sites one by one. For each cell, we first find out the nearest violation-free (meet all the hard and soft constraints) site in the same rows as the cell, if the distance between the cell and the nearest violation-free site is less than *amm*, then the cell will be placed to this site; otherwise, the cell will be marked as an illegal cell. In our implementation, *amm* is set as two placement sites.

3.3.3 Illegal Cell Handling. For each illegal cell c_i, we first extract blank intervals according to the placed cells positions and the cell height of c_i. Then, we traverse all blank intervals within a certain search range to find out the candidate intervals *cand* that can contain the cell c_i by pushing cells to the left or right. If the candidate set *cand* is not empty, then we select the interval with the lowest movement and move the cell to make the placement legal; otherwise, we enlarge the search range and ignore the pin short/access issues caused by vertical pre-routed wires until a feasible solution is found. After the above operations, all cells are placed in the corresponding subchip legally.

3.4 Technology-Aware Refinement

Since we place illegal cells one by one, there may be excessive movements of some cells, especially the cells being placed near the end of legalization. The maximum movement and the total movement may be further reduced if the cell order and cell assignment are changed. Therefore, in this subsection, we first present a cell swapping method to reduce the maximum movement, and then a weighted bipartite graph is constructed to further reduce the total movement, while ensuring that the maximum movement does not increase. Note that, during the refinement, legality of the placement is always maintained and the number of technology constraint violations are not increased.

3.4.1 Maximum Movement Optimization. We optimize the maximum movement by iteratively cell swapping as follows. First, the cell c_i with the largest movement is selected, and we take the rectangle $R_i = (\frac{x'_i+x_i-m_i}{2}, \frac{y'_i+y_i-m_i}{2}, \frac{x'_i+x_i+m_i}{2}, \frac{y'_i+y_i+m_i}{2})$ as the search area, where (x'_i, y'_i) is the global placement position, (x_i, y_i) the current position, and $m_i = |x_i - x'_i| + |y_i - y'_i|$ is the current movement of cell c_i. Then, the candidate cells located in the search area are collected, and each candidate cell satisfies the following conditions after swapping with the cells c_i: (1) the placement is still legal; (2) the number of technology constraint violations do not increase; and (3) the maximum movement is reduced. Finally, the candidate cell with the largest decrease in the maximum movement is selected to swap with cell c_i. The cell swapping process stops if the cell with maximum movement does not have any candidate cell to exchange.

3.4.2 Total Movement Optimization. After the maximum movement optimization, we perform a bipartite matching in each local area to further reduce the total movement, while ensuring that the maximum movement does not increase. We first divide the placement rectangle into $M \times N$ bins (the size of each bin is defined as 8×8 row heights in our implementation). Then a bipartite matching is performed to reduce the total movement for cells of the same type in each bin.

Let $C_T \subseteq C$ be the subset of cells with the same type in a bin. Let P_T be the set of current positions of the cells in C_T, and *mm* be the maximum movement of the cells in C_T. Then a bipartite graph $G = (C_T, P_T, E)$ is constructed, where the edge $(c_i, p_j) \in E$ if and only if the Manhattan distance $D_{i,j}$ between the global placement position of cell c_i and position p_j is not greater than *mm*. The problem is to find a perfect matching $PM \subseteq E$ such that the total cost $\sum_{(c_i,p_j)\in PM} D_{i,j}$ is minimized, which can be optimally solved by the Kuhn-Munkres algorithm [12].

Table 1: **Statistics of ICCAD 2017 CAD Contest Benchmarks.**

Benchmark	#S. Cell	#D. Cell	#T. Cell	#Q. Cell	Density
des_perf_1	112644	0	0	0	0.91
des_perf_a_md1	103589	4699	0	0	0.55
des_perf_a_md2	105030	1086	1086	1086	0.56
des_perf_b_md1	106782	5862	0	0	0.55
des_perf_b_md2	101908	6781	2260	1695	0.65
edit_dist_1_md1	118005	7994	2664	1998	0.67
edit_dist_a_md2	115066	7799	2599	1949	0.59
edit_dist_a_md3	119616	2599	2599	2599	0.57
fft_2_md2	28930	2117	705	529	0.83
fft_a_md2	27431	2018	672	504	0.32
fft_a_md3	28609	672	672	672	0.31
pci_bridge32_a_md1	26680	1792	597	448	0.50
pci_bridge32_a_md2	25239	2090	1194	994	0.58
pci_bridge32_b_md1	26134	1756	585	439	0.27
pci_bridge32_b_md2	28038	292	292	292	0.18
pci_bridge32_b_md3	27452	292	585	585	0.22

4 EXPERIMENTAL RESULTS

We implemented our legalization algorithm for the mixed-cell-height circuits with technology and fence region constraints in C++ programming language, and tested it on the benchmarks (including the hidden cases) of the 2017 ICCAD CAD Contest on mixed-cell-height standard cell Legalization [5]. Table 1 lists the benchmark statistics, where "#S. Cell", "#D. Cell", "#T. Cell", "#Q. Cell", and "Density" give the total numbers of single-, double-, triple-, quadruple-row height cells and the design density, respectively. The design sizes range from 28k to 131k with density from 18% to 91%.

4.1 Comparison with the State-of-the-Art Works

In the first experiment, we compared our legalization algorithm with the champion of the 2017 ICCAD CAD Contest [5] and the state-of-the-art work [10]. All the three algorithms were run on the same Linux machine with eight cores of 3.40GHz Intel Core CPU and 16GB memory. It should be noted that, the algorithm in [10] was implemented with multi-threading and we ran it with eight threads (the maximum number of threads allowed in the contest [5]), while the other two algorithms (including ours) were run in only single thread.

The score function in [10] is used to evaluate the quality of legalization results, which keeps the score function in the contest [5] as much as possible. The score function is

$$S = \left(1 + S_{hpwl} + \frac{N_p + N_e}{n}\right)\left(1 + \frac{M_{max}}{100}\right) S_{am}, \tag{4}$$

where S_{hpwl}, N_p, N_e, n and M_{max} are the increasing ratio in HPWL, the number of pin short/access violations, the number of edge spacing violations, the number of cells, and the maximum cell movement, respectively. The average cell movement S_{am} is defined as

$$S_{am} = \frac{1}{K}\sum_{i=1}^{K} M_{avg,i},$$

where cells are grouped into K sets based on their heights, and $M_{avg,i}$ is the average movement of cells with height equal to i rows [5]. The S_{am} and M_{max} are measured in terms of single row height.

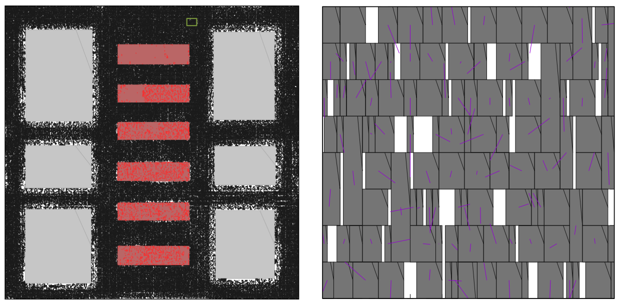
Table 2 shows the experimental results. In the table, the columns "Top1", "DAC'18", and "Ours" represent the corresponding results generated by the champion of the 2017 ICCAD CAD Contest [5], the work in [10], and our algorithm, and "HPWL" and "CPU" give the respective wirelength in meter and the runtimes in second. Compared with "Top1", our algorithm achieves 24% smaller average cell movement and 18% smaller maximum cell movement, as well as 35% smaller score in a comparable runtime. Compared with the state-of-the-art work "DAC'18" [10], our algorithm achieves 6% smaller average cell movement, 7% smaller maximum cell movement, and 7% smaller score as well as 69% smaller runtime. Specially, our algorithm does not generate any edge spacing violations and has the fewest pin short/access violations for all the benchmarks. Overall, the experimental results show that our algorithm achieves the best average cell movement and maximum cell movement, as well as significantly fewer technology constraint violations in comparable runtime.

4.2 Effectiveness of Individual Techniques

In the second experiment, we verified the effectiveness of the movement-aware cell reassignment and the technology-aware refinement. Table 3 lists the experimental results. In the table, the columns "w/o M." and "w/o T." give the corresponding results generated by our algorithm with the movement-aware cell reassignment being turned off and the technology-aware refinement being turned off, respectively.

As shown in the table, our algorithm with the movement-aware cell reassignment on average can achieve 3% smaller average movement and 35% smaller maximum movement. This is because the movement-aware cell reassignment reassigns disruptive cells in locally dense areas to locally sparse areas, and the overlapping superposition effects and excessive cell movement are relieved. In addition, the average movement and maximum movement can be reduced by 2% and 15% through the technology-aware refinement. The experimental results show the effectiveness of the movement-aware cell reassignment and the technology-aware refinement.

Fig. 6(a) shows the legalization result of the benchmark "edit_dist_a_md2" generated by our legalization algorithm, and Fig. 6(b) presents a partial layout.



(a) (b)

Figure 6: (a) Legalization result of the benchmark edit_dist_a_md2. (b) Partial layout of (a). Cells are in gray, and movement in purple.

5 CONCLUSIONS

In this paper, we have presented an effective and efficient legalization algorithm for mixed-cell-height circuit designs with technology

Table 2: Experimental Results.

Benchmark	S_{am}			M_{max}			HPWL(m)			N_p			N_e			Score S			CPU(s)		
	Top1	DAC'18	Ours	Top1	DAC'18	Ours	Top1	DAC'18	Ours	Top1	DAC'18	Ours	Top1	DAC'18	Ours	Top1	DAC'18	Ours	Top1	DAC'18	Ours
des_perf_1	0.710	0.903	0.807	7.7	8.4	6.6	1.30	1.35	1.33	11313	1815	1295	0	0	0	0.89	1.10	0.95	6.87	7.91	15.05
des_perf_a_md1	1.818	1.122	0.994	62.6	60.7	60.7	2.26	2.24	2.23	109	90	52	0	0	0	3.09	1.87	1.65	3.98	6.70	3.94
des_perf_a_md2	3.476	1.380	1.328	68.0	48.1	40.4	2.28	2.26	2.26	87	188	57	0	0	0	6.12	2.12	1.93	4.02	6.47	5.33
des_perf_b_md1	0.698	0.725	0.701	9.0	10.0	9.0	2.16	2.16	2.16	269	168	122	0	0	0	0.78	0.82	0.78	3.35	5.95	3.67
des_perf_b_md2	0.655	0.718	0.655	20.0	23.3	19.4	2.19	2.20	2.20	12	26	0	0	0	0	0.81	0.91	0.80	3.04	6.21	3.11
edit_dist_1_md1	0.798	0.752	0.765	7.9	5.7	7.8	4.09	4.09	4.10	0	45	0	0	0	0	0.88	0.81	0.84	3.77	6.34	3.84
edit_dist_a_md2	0.646	0.697	0.639	16.4	16.4	16.4	5.18	5.18	5.18	69	42	0	0	0	0	0.76	0.82	0.75	3.48	7.78	4.45
edit_dist_a_md3	0.901	0.837	0.838	28.0	31.4	23.3	5.46	5.45	5.48	158	1342	131	0	0	0	1.18	1.14	1.06	32.75	10.39	13.14
fft_2_md2	0.675	0.905	0.837	6.6	7.1	7.2	0.49	0.51	0.52	4139	196	0	5980	0	0	1.01	1.11	1.05	0.86	1.82	1.15
fft_a_md2	0.566	0.631	0.568	34.3	34.3	34.3	1.11	1.11	1.11	84	4	0	0	0	0	0.77	0.86	0.77	0.75	1.60	0.85
fft_a_md3	0.536	0.605	0.538	11.0	11.3	11.0	0.97	0.96	0.97	90	2	0	0	0	0	0.61	0.68	0.61	0.78	1.86	0.88
pci_bridge32_a_md1	0.696	0.712	0.664	42.6	45.9	42.6	0.47	0.48	0.47	30	25	0	0	0	0	1.03	1.09	0.98	0.83	1.76	1.10
pci_bridge32_a_md2	0.898	0.872	0.894	27.2	18.1	18.1	0.59	0.60	0.60	2243	183	171	0	0	0	1.29	1.10	1.13	1.34	1.50	2.61
pci_bridge32_b_md1	1.064	0.853	0.824	87.7	51.4	51.4	0.69	0.68	0.69	16	3	0	0	0	0	2.07	1.34	1.29	0.93	1.72	0.91
pci_bridge32_b_md2	1.084	0.785	0.791	72.3	61.7	54.6	0.60	0.59	0.59	26	5	0	0	0	0	1.93	1.31	1.26	0.82	4.26	0.92
pci_bridge32_b_md3	1.910	1.031	1.024	68.2	49.8	49.8	0.62	0.61	0.61	33	38	9	0	0	0	3.39	1.61	1.60	1.12	2.15	1.35
Normalized	1.24	1.06	1.00	1.18	1.07	1.00	1.00	1.00	1.00							1.35	1.07	1.00	0.93	1.69	1.00

Table 3: Average and Maximum Movements Impact of Movement-aware cell Reassignment and Technology-aware Refinement.

Benchmark	S_{am}			M_{max}		
	w/o M.	w/o T.	Ours	w/o M.	w/o T.	Ours
des_perf_1	0.869	0.853	0.807	19	12.4	6.6
des_perf_a_md1	1.047	1.023	0.994	60.7	60.7	60.7
des_perf_a_md2	1.371	1.354	1.328	40.4	40.4	40.4
des_perf_b_md1	0.723	0.711	0.701	20	11.6	9.0
des_perf_b_md2	0.663	0.662	0.655	20	20	19.4
edit_dist_1_md1	0.781	0.778	0.765	14.9	9.4	7.8
edit_dist_a_md2	0.653	0.647	0.639	16.4	16.4	16.4
edit_dist_a_md3	0.883	0.861	0.838	26.8	35.7	23.3
fft_2_md2	0.880	0.857	0.837	17.5	10.5	7.2
fft_a_md2	0.573	0.572	0.568	34.3	34.3	34.3
fft_a_md3	0.548	0.542	0.538	11.0	11.0	11.0
pci_bridge32_a_md1	0.672	0.671	0.664	42.6	42.6	42.6
pci_bridge32_a_md2	0.927	0.912	0.894	18.2	18.2	18.1
pci_bridge32_b_md1	0.837	0.833	0.824	51.8	51.4	51.4
pci_bridge32_b_md2	0.816	0.813	0.791	54.6	54.6	54.6
pci_bridge32_b_md3	1.036	1.039	1.024	51	49.8	49.8
Normalized	1.03	1.02	1.00	1.35	1.15	1.00

and region constraints. We first presented the fence region handling technique to unify the fence regions and the default ones. To obtain a desired cell assignment, we then proposed the movement-aware cell reassignment method by iteratively reassigning cells in locally dense areas to their desired rows. After cell reassignment, the technology-aware legalization was presented to remove cell overlaps and align each cell to placement site, while satisfying the technology constraints. Finally, the technology-aware refinement was applied to further reduce the average and maximum cell movements without increasing the technology constraints violations. Compared with the champion of the ICCAD 2017 CAD Contest and the recent state-of-the-art work, experimental results shown that our algorithm achieved the best average cell movement and maximum cell movement, and significantly fewer technology constraints violations in a comparable runtime.

REFERENCES

[1] Z.-Z. Bai. Modulus-based matrix splitting iteration methods for linear complementarity problems. *Numerical Linear Algebra with Applications*, 17(6):917–933, 2010.

[2] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis. ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. In *Proceedings of International Symposium on Physical Design*, 2015.

[3] J. Chen, Z. Zhu, W. Zhu, and Y.-W. Chang. Toward optimal legalization for mixed-cell-height circuit designs. In *Proceedings of ACM/IEEE Design Automation Conference*, 2017.

[4] W.-K. Chow, C.-W. Pui, and E. F. Y. Young. Legalization algorithm for multiple-row height standard cell design. In *Proceedings of ACM/IEEE Design Automation Conference*, 2016.

[5] N. K. Darav, I. S. Bustany, A. Kennings, and R. Mamidi. ICCAD-2017 CAD contest in multi-deck standard cell legalization and benchmarks. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2017.

[6] N. K. Darav, A. Kennings, A. F. Tabrizi, D. Westwick, and L. Behjat. Eh?Placer: A high-performance modern technology-driven placer. *Acm Transactions on Design Automation of Electronic Systems*, 21(3):37:1–37:27, April 2016.

[7] C.-C. Huang, C.-H. Chiou, K.-H. Tseng, and Y.-W. Chang. Detailed-routing-driven analytical standard-cell placement. In *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, pages 378–383, 2015.

[8] C.-C. Huang, H.-Y. Lee, B.-Q. Lin, S.-W. Yang, C.-H. Chang, S.-T. Chen, Y.-W. Chang, T.-C. Chen, and I. Bustany. NTUplace4dr: a detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, June 2017.

[9] A. Kennings, N. K. Darav, and L. Behjat. Detailed placement accounting for technology constraints. In *Proceedings of IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, 2015.

[10] H. Li, W.-K. Chow, G. Chen, E. F. Y. Young, and B. Yu. Routability-driven and fence-aware legalization for mixed-cell-height circuits. In *Proceedings of ACM/IEEE Design Automation Conference*, 2018.

[11] Y. Lin, B. Yu, X. Xu, J.-R. Gao, N. Viswanathan, W.-H. Liu, Z. Li, C. J. Alpert, and D. Z. Pan. MrDP: multiple-row detailed placement of heterogeneous-sized cells for advanced nodes. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 7:1–7:8, 2016.

[12] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[13] D. K. Smith. Network flows: Theory, algorithms, and applications. *Journal of the Operational Research Society*, 45(11):1340–1340, 1994.

[14] C.-H. Wang, Y.-Y. Wu, J. Chen, Y.-W. Chang, S.-Y. Kuo, W. Zhu, and G. Fan. An effective legalization algorithm for mixed-cell-height standard cells. In *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, 2017.

[15] C.-K. Wang, C.-C. Huang, S. S.-Y. Liu, C.-Y. Chin, S.-T. Hu, W.-C. Wu, and H.-M. Chen. Closing the gap between global and detailed placement: Techniques for improving routability. In *Proceedings of International Symposium on Physical Design*, pages 149–156, 2015.

[16] G. Wu and C. Chu. Detailed placement algorithm for VLSI design with double-row height standard cells. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(9):1569–1573, September 2016.