Routability-Driven and Fence-Aware Legalization for Mixed-Cell-Height Circuits

Haocheng Li
CSE Department, CUHK
hcli@cse.cuhk.edu.hk

Wing-Kai Chow
Cadence Design System Inc.
wkchow@cadence.com

Gengjie Chen
CSE Department, CUHK
gjchen@cse.cuhk.edu.hk

Evangeline F. Y. Young
CSE Department, CUHK
fyyoung@cse.cuhk.edu.hk

Bei Yu
CSE Department, CUHK
byu@cse.cuhk.edu.hk

ABSTRACT

Placement is one of the most critical stages in the physical synthesis flow. Circuits with increasing numbers of cells of multi-row height have brought challenges to traditional placers on efficiency and effectiveness. Furthermore, constraints on fence region and routability (e.g., edge spacing, pin access/short) should be considered, besides providing an overlap-free solution close to the global placement (GP) solution and fulfilling the power and ground (P/G) alignments. In this paper, we propose a legalization method for mixed-cell-height circuits by a window-based cell insertion technique and two post-processing network-flow-based optimizations. Compared with the champion of the IC/CAD 2017 Contest, our algorithm achieves 18% and 12% less average and maximum displacement respectively as well as significantly fewer routability violations. Comparing our algorithm with the state-of-the-art algorithms on this problem, there is a 9% improvement in total displacement with 20% less running time.

1 INTRODUCTION

With the continuous shrinking of the semiconductor feature size to 7 nm [1], the number of in-cell tracks diminishes significantly and the internal routability within a standard cell becomes inadequate. In sub-10 nm technology nodes, there are cells of just six-track-high [2], which results in difficulties in designing complex standard cells like multiplexers [3] and multi-bit flip-flops [4] with high driving strength. To achieve ascending performance and efficiency, complex cells are now designed with multi-row-height [5], while simple cells remain single-row-height for area efficiency.

Previous works on legalization algorithm for mixed-cell-height circuits can be categorized into three types [6]. (1) One honors the horizontal cell order of GP. For example, Wang *et al.* [7] extend the dynamic-programming-based single-row-cell legalizer Abacus [8] for multi-row cells. They legalize cells from left to right and evaluate the cost of placing a cell in different rows while considering the dead-space created. Chen *et al.* [9] formulate the legalization problem as a quadratic program with a quadratic displacement objective and transform it into a linear complementary problem (LCP) by the Karush-Kuhn-Tucker (KKT) conditions. It should be noted there are several recent ordered legalization algorithms under single-row-cell scenario (e.g., [10, 11]). However, as a strong and unnecessary constraint, maintaining the cell order of GP highly restricts the solution space, thus it may result in poor results especially for a dense design. (2) The second type is free from the artificial restriction on cell order. Chow *et al.* [12] propose a multi-row local legalization (MLL) algorithm, which explores different row assignments and cell order in a window around the GP location of each cell. It sequentially legalizes cells by maintaining the row assignments and relative order of the previously legalized cells. For each new cell, different row and order assignments are attempted, where the total displacement is minimized by shifting cells horizontally. Its major limitation is that the minimized displacement is w.r.t the current locations of the cells, which can be accumulated to large displacements from GP locations after many iterations. MrDP [13] proposes a wirelength-driven legalization based on a chain move scheme and extends a dual min-cost flow (MCF) method [14] from single- to multi-row cells for post-refinement. A potential problem is that an objective of half-perimeter wire-length (HPWL) instead of displacement in legalization may disturb some other metrics optimized in GP. (3) There are some recent works that legalize mixed-cell-height circuits with additional constraints like IR drops [2] and minimum implant area (MIA) [15]. However, none of the previous works solve the problem comprehensively. Some of them target at minimizing HPWL while some focus on the total displacement, but none of them simultaneously handle other important measures and constraints like the existence of fence region and routability issues which include pin access, pin short and edge spacing.

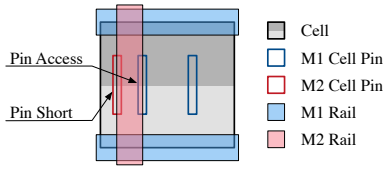


Figure 1: Pin access and pin short.

In this paper, we present a fast and high-quality legalization framework for standard cells with mixed cell heights, which outperforms the state-of-the-art under the displacement objective. Our major contributions are as follows.

- We develop a mixed-cell-height circuit legalizer optimizing the maximum and average displacement with constraints on routability and fences.
- We devise a method called multi-row global legalization (MGL) that inserts a cell optimally into a window, minimizing an average displacement of all the cells in the region from their GP positions.
- A bipartite graph matching is devised to minimize the maximum displacement among a group of cells that can exchange their positions without creating additional violations.
- We extend the min-cost flow (MCF) formulation of the fixed-row-and-order problem to one that optimizes a weighted sum of the maximum and average displacement, with range constraints on the cell movements to avoid pin short and pin access violations.

DAC '18, June 24–29, 2018, San Francisco, CA, USA

ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196107

The rest of the paper is organized as follows. Section 2 illustrates the problem formulation and constraints. Section 3 provides a detailed explanation of our proposed techniques. Section 4 verifies the effectiveness of our approach, followed by a conclusion in Section 5.

2 PROBLEM FORMULATION

Given a set of m multi-row height cells $C = \{c_1, c_2, \cdots, c_m\}$. The cell height and displacement are measured in terms of the number of single row heights. Let H be the largest cell height, $C_h \subseteq C$ be then set of cells whose height is h. The problem is to place each cell c_i from global placement (GP) (x'_i, y'_i) into (x_i, y_i) with a corresponding displacement:

$$\delta_i = \delta_{x\,i} + \delta_{y\,i} = |x_i - x'_i| + |y_i - y'_i|, \tag{1}$$

such that the maximum and average displacement is minimized. Here, the average displacement S_{am} is weighted by the number of cells of the same height:

$$S_{am} = \frac{1}{H}\sum_{h=1}^{H}\frac{1}{|C_h|}\sum_{c_i \in C_h}\delta_i, \tag{2}$$

which is the metric used in the IC/CAD 2017 Contest [16].

Besides, cells should be overlap-free and aligned to placement sites of the chip. The power and ground (P/G) alignment and the fence region constraint are treated as hard constraints: (1) Cells with even cell heights must be placed in alternate rows with aligned P/G rails [12]; (2) Cells assigned to a fence region must be placed inside the fence boundary [17]. Note that cells of odd cell heights have no restriction on the row assignments because they can be flipped to align with the P/G rails.

The routability constraints including edge spacing and pin access/short are considered as soft constraints [18]:

- A minimum spacing is required between any two cell edges;
- Signal pins of cells should not be short or inaccessible due to the P/G grids and IO pins.

Note that a signal pin on metal layer k is short if it overlaps with a P/G rail or an IO pin on metal layer k; it is inaccessible if it overlaps with a P/G rail or an IO pin on metal layer $k+1$ [19]. As shown in Figure 1, the left pin on metal layer one (M1) has pin access problem with the rail on metal layer two (M2), and the M2 pin is short with the M2 rail. In modern chip design, the P/G rails are usually regular grids running horizontally and vertically in alternate metal layers [16].

3 ALGORITHMS

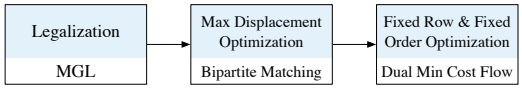


Figure 2: The proposed legalization flow.

The overall algorithmic flow is illustrated in Figure 2, which consists of three stages. (1) Given a GP solution with multi-row height cells, we will first legalize it by MGL which inserts the cells sequentially into the placement region. Note that a cell may belong to a specific fence region. Cells that do not belong to any fence regions should be placed in the *default fence* region which is the region outside all the given fence regions. (2) Next, the maximum displacement is optimized by swapping cells of the same type in the same fence region while maintaining the average displacement. (3) Finally, keeping the rows and cell order unchanged, the average and maximum displacement is further optimized. Details of these three major steps will be explained in the following sub-sections.

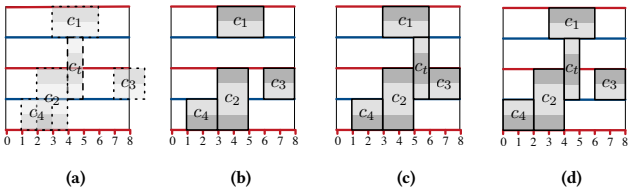


Figure 3: Comparison between MLL and MGL: (a) GP; (b) Four cells legalized; Final results of minimizing the total displacement w.r.t. (c) current locations (MLL) and (d) GP locations (MGL).

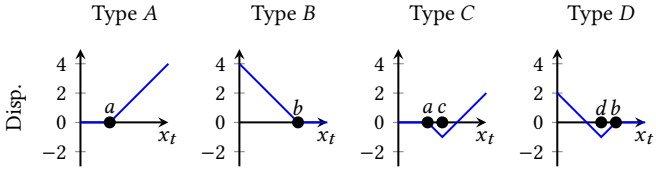


Figure 4: Four types of displacement curves.

3.1 Legalization

In this section, we will introduce the multi-row global legalization (MGL) method which legalizes cells sequentially to minimize the average and maximum displacement from the given GP positions.

MGL legalizes cells sequentially and is inspired by MLL [12]. Different from MLL that calculates displacement based on the current cell locations and can eventually accumulate a large displacement w.r.t. GP locations, MGL minimizes the displacement from GP locations directly. Figure 3 illustrates an example, where the given GP positions are shown in Figure 3(a). Suppose cells $c_1 - c_4$ are legalized before target cell c_t is inserted as in Figure 3(b), which already has a total displacement of two. In Figure 3(c), MLL optimizes the total displacement w.r.t. current locations and achieves a value of one. However, the total displacement from GP position is actually three. Figure 3(d) shows the result with minimized total displacement from GP positions (i.e., two) produced by MGL.

Algorithm 1 shows the flow of MGL. When legalizing a *target cell* c_t, a *window* r_t around its GP location will be considered. Meanwhile, legalized cells that lie completely within r_t are referred as *local cells*, which can be shifted for legalizing c_t. In MGL, the row and order assignment of local cells are fixed, but that of c_t will be enumerated and evaluated. With local cells' row and their relative order fixed, inserting c_t with height h implies that we need to place c_t in some gaps between the legalized cells in h consecutive rows. A combination of those gaps for inserting c_t is an *insertion point*. For a given c_t and r_t, MGL first obtains all the legal insertion points by using the enumerating method in [12] (line 1). It then calculates the optimal displacement cost of all the insertion points (lines 2-11). The displacement curve, which represents the cost of each insertion point with varied x-coordinate of c_t, can be constructed by adding up the all displacement curves of the local cells and c_t. The construction of the curves will be explained in more details later. The best position to insert c_t with an insertion point is the position with the lowest cost on the displacement curve. After inserting c_t at the position with the lowest cost, local cells will be shifted to the left or right to legalize the placement (lines 12-13). The windows size will increase if c_t cannot be inserted.

In MLL, when a valid insertion point p is considered, there are only two types of displacement curves for any local cell as illustrated by type A and B in Figure 4 in which the horizontal axis is the x-position of the target cell and the vertical axis is the displacement contributed by the local cell. The curves are of these shapes since we are measuring

the distance from the original positions of the local cells before the target cell is inserted. Cells on the right (left) of p in the window have displacement curves of type A (B) because they may be pushed to the right (left) by placing the target cell at different positions. The turning points of these curves are called *critical positions* [12]. Since there are only these two types of curves in MLL, the optimal position to place the target cell can be obtained efficiently by finding the median of all these critical positions.

Algorithm 1 MGL

Input: Window r_t, GP position (x'_t, y'_t) of target cell c_t.
Output: Legal positions of c_t and local cells.

```
Find candidate insertion points {p_i} in r_t;
for all p_i ∈ {p_i} do
    for all breakpoint b do
        Store (x_b, left slope of b, right slope of b) in points;
    end for
    Sort points;
    Construct total displacement curve of c_t and local cells;
    d_i ← optimal displacement;
    x_t^i ← optimal x-coordinate;
    y_t^i ← y-coordinate of p_i;
end for
j ← arg min_i d_i;
Place c_t at (x_t^j, y_t^j) and spread local cells;
```

In MGL, the scenario is more complicated since the displacement is counted w.r.t. the given GP position. There are two more types of displacement curves as illustrated by types $A-D$ in Figure 4. Considering the local cells on the right of a valid insertion point p, there are two possible types of curves A and C. Cells with their GP positions at or on the left of their current positions have displacement curves type A because the target cell will only push them further to the right from their GP positions. For cells with their GP positions on the right of their current positions will have displacement curve type C. The turning points on these curves are either critical positions as in MLL (those labeled by a) or positions computable from the GP positions of the local cells (those labeled by c). We call all these turning points *breakpoints*. Similarly, for local cells which are on the left of the valid insertion points, their displacement curves will be of types B and D. Theorem 1 states that the final displacement curve is convex if the local cells are originally at their optimal positions w.r.t. their GP positions, before the target cell is inserted.

Theorem 1. *Consider a window W containing a target position (x'_t, y'_t), a set S of local cells lying completely inside W and a target cell c_t to be inserted into W. If all the cells in S are originally placed at optimal positions (total displacement is the smallest under the fixed row & fixed order constraint) w.r.t their GP positions, the displacement curve, where the x-axis is the position of the target cell x_t, obtained by adding up the displacement curves of all the cells in S is convex and piecewise linear.*

The proof is skipped here due to the lack of space. The pre-condition of having the local cells at optimal positions w.r.t. their GP positions would require running a min-cost flow (MCF) before invoking MGL that will lengthen the running time. Therefore, in our implementation, we will compute the cost at each breakpoint to find the optimal position. Since the number of breakpoints is linear with the number of local cells, so the optimal positions can be found very efficiently.

3.2 Maximum Displacement Optimization

In this section, we will present the maximum displacement optimization method in a legal placement. Recall that in MGL, each cell is processed sequentially and it will then be fixed to a segment once placed. The maximum displacement can be further reduced if the row assignments can be changed, especially for the cells being placed near the end of MGL. It is inevitable to place them with large displacements if the regions around their GP locations are dense. Figure 6(a) shows the displacement of a cell type in a fence region. Each rectangle represents a cell. Red cells are of the same type and gray cells are of other types. The red lines connect cells to their corresponding GP positions. We can see that some cells are placed to even tens of rows away from their GP positions.

To reduce the maximum displacement without creating violations in the legal placement, we perform a min-cost bipartite matching to optimize the maximum displacement for each cell type in each fence region. Given a bipartite graph $G = (C_T, P_T, C_T \times P_T)$ on a subset of cells of the same type $C_T \subseteq C$ and the current positions $P_T \subseteq P$ of the cells in C_T, any cell $c_i \in C_T$ can take up the positions $p_j = (x_j, y_j)$ of another cell c_j to minimize the maximum displacement without creating any violations. The problem is to find a perfect matching $S \subseteq C_T \times P_T$ between cells and positions with the minimum total cost $\sum_{(c_i,p_j)\in S} D_{i,j}$, where $D_{i,j} = \phi(|x_j - x'_i| + |y_j - y'_i|)$ and $\phi(\delta)$ is defined as a strictly increasing function such that it is linear when δ is small to preserve the average displacement. After a certain threshold of δ, ϕ will increase rapidly in order to discourage large displacement. Here, we have:

$$\phi(\delta) = \begin{cases} \delta, & \delta \le \delta_0, \\ \frac{\delta^5}{\delta_0^4}, & \text{otherwise}, \end{cases} \tag{3}$$

where δ_0 is the tolerable maximum displacement threshold. This min-cost perfect matching problem can be optimally solved by formulating as an MCF problem [20].

There are previous works that use bipartite matching in detailed placement to minimize HPWL [21] but only those cells on "independent" nets can be optimized simultaneously. Here the cost function ϕ is defined in such a way that both the average and maximum displacement are handled and all the cells of the same type can be optimized.

3.3 Fixed Row & Fixed Order Optimization

After the matching-based maximum displacement optimization, we perform a final post-processing refinement to further reduce the maximum and average displacement by shifting the cells locally without changing the cell order and row assignments. Taking the objective of the total displacement as an example, given a set of m multi-row height cells $C = \{c_i\}$, the problem can be formulated as a linear program (LP):

$$\min_{x_i} \sum_i n_i \delta_{x\,i} \tag{4}$$

$$\text{s.t. } x_i + w_i \le x_j, \qquad \forall (i,j) \in E, \tag{4a}$$

$$l_i \le x_i \le r_i, \qquad \forall c_i \in C, \tag{4b}$$

where n_i is the weight on the x-displacement $\delta_{x\,i}$ of cell c_i, w_i is the width of c_i, l_i and r_i are the left and right boundary of the segment containing c_i, E is the set of neighboring pairs where $(i,j) \in E$ if and only if c_i is the left neighbor of c_j on some rows.

The LP in (4) can be converted to a dual MCF problem and effectively solved [13, 14]. Compared to the MCF formulation in [13], our transformation to MCF has three strengths. (1) There are significantly fewer vertices in the flow network, which is more efficient. (2) The maximum and average displacement are optimized simultaneously. (3) Weight n_i is set according to (2), while it is ignored (i.e., all one) in [13].

To obtain the MCF formulation, we first split the x-displacement $\delta_{x\,i}$ in (4) to a pair of variables x_i^- and x_i^+ and achieve (5).

In (5c) and (5d), $\{x_i^l\}$ and $\{x_i^r\}$ are auxiliary variables, C_L is the set of left-most cells in at least one of the segments and C_R is the set of

$$\max_{x_i, x_i^-, x_i^+, x_i^l, x_i^r} \sum_i n_i(x_i^- - x_i^+) \tag{5}$$

$$\text{s.t. } x_i^- \le 0, x_i - x_i' \le x_i^+, \quad \forall c_i \in C, \tag{5a}$$

$$x_i - x_j \le -w_i, \quad \forall (i,j) \in E, \tag{5b}$$

$$0 \le x_i^l \le x_i - l_i, \quad \forall c_i \in C_L, \tag{5c}$$

$$x_i - r_i \le x_i^r \le 0, \quad \forall c_i \in C_R. \tag{5d}$$

right-most cells in at least one of the segments. The dual LP of (5) is:

$$\min_f \ \mathcal{Q} = \sum_i \left(x_i'(f_i^+ - f_i^-) - l_i f_i^l + r_i f_i^r\right) - \sum_{(i,j)\in E} w_i f_{ij} \tag{6}$$

$$\text{s.t. } \mathcal{F}_i = f_i^+ - f_i^- + f_i^r - f_i^l + \sum_{j:(i,j)\in E} f_{ij} - \sum_{k:(k,i)\in E} f_{ki} = 0, \quad \forall c_i \in C, \tag{6a}$$

$$0 \le f_i^+, f_i^- \le n_i, \quad \forall c_i \in C, \tag{6b}$$

$$f_i^l, f_i^r \ge 0, \quad \forall c_i \in C, \tag{6c}$$

$$f_i^l = 0, \quad \forall c_i \in C - C_L, \tag{6d}$$

$$f_i^r = 0, \quad \forall c_i \in C - C_R, \tag{6e}$$

$$f_{ij} \ge 0, \quad \forall (i,j) \in E. \tag{6f}$$

Summing up constraints (6a) together, we further have:

$$\mathcal{F}_z = -\sum_i \mathcal{F}_i = 0, \tag{7}$$

which implies the flow conservation in the flow graph. By considering (6) and (7), an MCF problem is formulated where each variable in (5) corresponds to a vertex in the graph while the constraints in (5) are illustrated by directed edges and dual variables $\{f\} = \{f_i^-\} \cup \{f_i^+\} \cup \{f_{ij}\} \cup \{f_i^l\} \cup \{f_i^r\}$. (7) is achieved by the flow conservation at an additional auxiliary vertex v_z. Note that each of the auxiliary vertices $\{v_i^-\}$, $\{v_i^+\}$, $\{v_i^l\}$, $\{v_i^r\}$ connects only two edges which can be combined to form one edge. Hence, they can be eliminated. Note that this is a MCF problem with $m+1$ vertices and $2m + |C_L| + |C_R| + |E|$ edges, while the MCF in [13] has $3m+2$ vertices and $6m + |E|$ edges. Our formulation is simpler and thus can be solved more efficiently.

3.3.1 Extension Considering Maximum Displacement. The formulation above optimizes the total displacement. To consider the maximum displacement, we further introduce a pair of auxiliary variables δ^-, δ^+ whose absolute values represent the largest displacement of the cells to the left and to the right of the corresponding GP position. Thus, we extend (5) to consider a weighted sum as follows:

$$\max_{\delta^-, \delta^+, x_i, x_i^-, x_i^+, x_i^l, x_i^r} \ n_0(\delta^- - \delta^+) + \sum_i n_i(x_i^- - x_i^+) \tag{8}$$

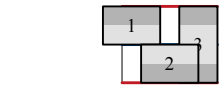
$$\text{s.t. } \delta^- \le \min\{0, x_i - x_i'\} - \delta_{y_i}, \quad \forall c_i \in C, \tag{8a}$$

$$\delta^+ \ge \max\{0, x_i - x_i'\} + \delta_{y_i}, \quad \forall c_i \in C, \tag{8b}$$

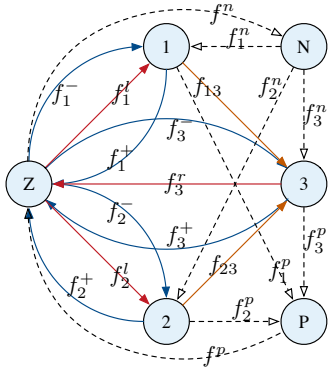
$$(5a) - (5d),$$

where n_0 can be tuned to balance the maximum and the average displacement in the objective function and δ_{y_i} is the y-displacement of cell c_i, which are constants since the row assignments will be preserved in this step. The dual LP is in (9), where f^p, f^n, $\{f_i^p\}$ and $\{f_i^n\}$ are auxiliary variables for handling the maximum displacement.

Figure 5(a) shows an example. Cells c_1 and c_2 are single-row while cell c_3 is double-row. The corresponding flow graph is shown in Figure 5(b). Vertices v_z, v_n and v_p are auxiliary nodes while each of the other nodes represents a cell in Figure 5(a). The solid straight edges from v_z (e.g., f_1^l) represent the flows for the constraints of the left boundary. The solid straight edge to v_z (there is only f_3^r in this toy case) represents the flow for the constraints of the right boundary. The other solid straight edges (e.g., f_{13}) illustrate the flows for the constraints between neighbouring



(a) GP



(b) Flow network

Flow	Cap	Cost
f_1^-	1/4	1
f_1^+	1/4	-1
f_2^-	1/4	-1
f_2^+	1/4	1
f_3^-	1/2	-3
f_3^+	1/2	3
f_1^l	∞	0
f_2^l	∞	0
f_3^r	∞	5
f_{13}	∞	-3
f_{23}	∞	-3
f_1^n	∞	1
f_1^p	∞	-1
f_2^n	∞	-1
f_2^p	∞	1
f_3^n	∞	-3
f_3^p	∞	3
f^n	1/50	0
f^p	1/50	0

(c) Edge capacity and cost

Figure 5: Example of fixed row and fixed order optimization.

$$\min_f \ \mathcal{Q} + (f^p + f^n)\max_i \delta_{y_i} + \sum_i \left(x_i'(f_i^p - f_i^n) - \delta_{y_i}(f_i^p + f_i^n)\right) \tag{9}$$

$$\text{s.t. } \mathcal{F}_i + f_i^p - f_i^n = 0, f_i^p, f_i^n \ge 0, \quad \forall c_i \in C, \tag{9a}$$

$$\mathcal{F}_z + f^n - f^p = 0, 0 \le f^p, f^n \le n_0, \tag{9b}$$

$$f^p - \sum_i f_i^p = 0, f^n - \sum_i f_i^n = 0, \tag{9c}$$

$$(6b) - (6f).$$

cells. The solid curly edges (e.g., f_i^-) represent the flows formulating the absolute value. The dotted edges (e.g., f^n) represent the flows for the formulation of the maximum displacement. The capacity and cost of the edges are shown in the table on the right of Figure 5.

In this work, we deploy a network simplex algorithm with first eligible pivot rule to solve the MCF problem. The worst case complexity is $\mathcal{O}(nm^2QU)$ [20], where n and m denote the number of nodes and arcs in the flow network respectively, U denotes the maximum arc capacities and Q denotes the largest arc cost.

3.4 Routability-Driven Refinement

Edge spacing rules define the minimum distances between different types of cells. The method in Section 3.2 will not create violations to any edge spacing rules because only cells of the same type will replace each other in the bipartite matching and all pairs of consecutive cell edges will remain unchanged. For the MGL and the fixed row and fixed order optimization, fillers will be inserted for correct edge spacing when calculating the width of the cell on the left of the insertion point.

Pin access and pin short violations are caused by signal pins of cells overlapping with P/G rails or IO pins on the same layer (called pin short) or on the next upper layer (called pin access). The violations can be divided in three types: overlaps with horizontal rails, with vertical rails and with IO pins. The method in Section 3.2 will not create pin access or pin short violations. In MGL, these three types of violations are considered separately. If an insertion point has violation with a horizontal rail, it will not be considered as a valid insertion point. While evaluating an insertion point, the optimal position is chosen accordingly to the displacement curve. If there is violation with a vertical rail,

Table 1: Comparison between Our Algorithm and the Champion in IC/CAD 2017 Contest

Benchmark	#Cells of Different Heights				Density	Avg. Disp.		Max. Disp.		HPWL (+e9)		Pin Access		Edge Space		Score S		Runtime (s)	
	1	2	3	4		1st	Ours	1st	Ours	1st	Ours	1st	Ours	1st	Ours	1st	Ours	1st	Ours
des_perf_1	112644	0	0	0	90.6%	0.710	0.903	7.7	8.4	1.30	1.35	11313	1815	0	0	0.89	1.10	9.548	10.878
des_perf_a_md1	103589	4699	0	0	55.1%	1.818	1.122	62.6	60.7	2.26	2.24	109	90	0	0	3.09	1.87	5.461	9.954
des_perf_a_md2	105030	1086	1086	1086	55.9%	3.476	1.380	68.0	48.1	2.28	2.26	87	188	0	0	6.12	2.12	5.503	9.369
des_perf_b_md1	106782	5862	0	0	55.0%	0.698	0.725	9.0	10.0	2.16	2.16	269	168	0	0	0.78	0.82	4.466	8.273
des_perf_b_md2	101908	6781	2260	1695	64.7%	0.655	0.718	20.0	23.3	2.19	2.20	12	26	0	0	0.81	0.91	4.159	9.098
edit_dist_1_md1	118005	7994	2664	1998	67.4%	0.798	0.752	7.9	5.7	4.09	4.09	0	45	0	0	0.88	0.81	5.166	9.319
edit_dist_a_md2	115066	7799	2599	1949	59.4%	0.646	0.697	16.4	16.4	5.18	5.18	69	42	0	0	0.76	0.82	4.306	9.808
edit_dist_a_md3	119616	2599	2599	2599	57.2%	0.901	0.837	28.0	31.4	5.46	5.45	158	1342	0	0	1.18	1.14	44.343	11.512
fft_2_md2	28930	2117	705	529	82.7%	0.675	0.905	6.6	7.1	0.49	0.51	4139	196	5980	0	1.01	1.11	1.175	2.961
fft_a_md2	27431	2018	672	504	32.3%	0.566	0.631	34.3	34.3	1.11	1.11	84	4	0	0	0.77	0.86	0.991	2.334
fft_a_md3	28609	672	672	672	31.2%	0.536	0.605	11.0	11.3	0.97	0.96	90	2	0	0	0.61	0.68	1.023	2.227
pci_bridge32_a_md1	26680	1792	597	448	49.5%	0.696	0.712	42.6	45.9	0.47	0.48	30	25	0	0	1.03	1.09	1.105	2.488
pci_bridge32_a_md2	25239	2090	1194	994	57.7%	0.898	0.872	27.2	18.1	0.59	0.60	2243	183	0	0	1.29	1.10	1.892	2.442
pci_bridge32_b_md1	26134	1756	585	439	26.6%	1.064	0.853	87.7	51.4	0.69	0.68	16	3	0	0	2.07	1.34	1.262	2.268
pci_bridge32_b_md2	28038	292	292	292	18.3%	1.084	0.785	72.3	61.7	0.60	0.59	26	5	0	0	1.93	1.31	1.076	6.952
pci_bridge32_b_md3	27452	292	585	585	22.2%	1.910	1.031	68.2	49.8	0.62	0.61	33	38	0	0	3.39	1.61	1.336	2.677
Norm. Avg.						1.18	**1.00**	1.12	**1.00**	1.00	1.00	8.25	**1.00**			1.26	**1.00**	**0.72**	1.00

positions on the left or on the right will be considered until a least-displaced-position without any violation is found. For IO pins, penalties will be given to the insertion points which overlap with IO pins.

Furthermore, to avoid more pin access and pin short violations in the fixed row and fixed order optimization, the cells will be restricted to a feasible range defined by the intersection of the row segment and the P/G rails or IO pins. Thus, every cell has its left and right boundary constraints in the MCF, i.e., $C_L = C_R = C$. l_i and r_i in the formulation is the left and right boundaries of cell c_i's feasible range respectively.

3.5 Multi-Thread Implementation

To speed up the process, MGL is implemented with multi-threading. Local windows that do not overlap with each other can be processed simultaneous. A scheduling step decides which local windows can be processed at the same time. The *scheduler* maintains a list L_p containing the local windows under processing. For a local window that failed to have the target cell inserted, it will be expanded and pushed into a waiting list L_w. The scheduler will select local windows that do not overlap with any window under processing and pushes them into L_p.

Legalizers in the child threads process MGL whenever there is any local windows in the L_p and return results. Since the scheduler synchronizes all threads after each MGL iteration, the multi-thread implementation is deterministic once the capacity of the list L_p is determined.

4 EXPERIMENTAL RESULTS

We implemented the proposed legalization algorithm for mixed-cell-height circuits in C++ programming language. LEMON [20] is used as the MCF solver. We run all experiments on a 64-bit Linux machine with eight cores of Intel Xeon 2.1GHz CPUs and 64GB RAM.

In the first experiment, we compare with a binary from the first place of IC/CAD 2017 Contest [16] and the results are listed in Table 1. The second to sixth columns show some statistics of the benchmarks including number of cells of different heights and design density. Here, *density* is measured by the total free area over the total cell area. We adopt the score function in the contest as much as possible[1] to have a more comprehensive comparison:

$$S = \left(1 + S_{hpwl} + \frac{N_p + N_e}{m}\right)\left(1 + \frac{\max_i\{\delta_i\}}{\Delta}\right) S_{am}, \tag{10}$$

where S_{hpwl} is the increasing ratio in HPWL, N_p and N_e are the numbers of violations on pin access/short and edge spacing, m is the number of cells, δ_i and S_{am} are calculated by (1) and (2) and Δ is 100 [16]. Compared with the first place, our proposed algorithm achieves 18% smaller average displacement and 12% shorter maximum displacement. For routability-driven constraints, we have no edge spacing violations while the first place produces nearly six thousand such violations. We also have significantly fewer pin access and pin short violations. In terms of S, our purposed method has 26% improvement on average.

In the second experiment, we compare with some state-of-the-art placers [7, 9, 12]. The benchmarks are modified from the ISPD 2015 Contest [17] and provided by the authors of [12]. 10% of the cells were selected and converted to double height and half width. Listed in Table 2, We adapted our program to use total displacement as the objective function and ignored fences as well as routability-driven constraints. Note that the results of [7, 12] are improved ones reported in [9]. We can see that we have improved over the previous published works by 20%, 17% and 9% respectively in total displacement.

In the third experiment, we verify the effectiveness of the two post-processing stages. Figure 6 shows an example of the maximum displacement optimization in which red cells are of the same type while red lines connect cells to their corresponding GP positions. Cells with large displacement in Figure 6(a) are moved to closer locations in Figure 6(b). Table 3 lists the average displacement and maximum displacement before and after the post-processing stages. We can see that through the proposed optimization, the maximum displacement and the average displacement can be decreased by around 23% and 1%.

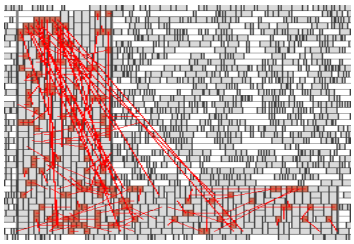
5 CONCLUSION

We presented a legalization method for mixed-cell-height circuits with consideration of routability constraints like pin access, pin shorts, edge spacing and fence regions. We proposed a multi-row global legalization (MGL) that minimizes the total displacement of the cells within a window towards their given global placement (GP) positions. We formulated and solved the maximum displacement optimization into by a min-cost flow (MCF). Finally, we formulated the fixed row and

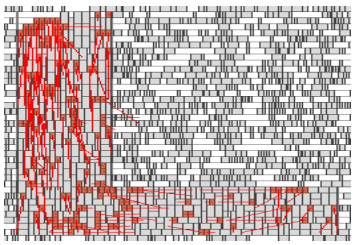
[1]The runtime scores are not included because they are measured w.r.t other teams. The penalties of maximum displacement and target utilization are not included because their exact definitions are not clear and a constant factor is not revealed.

Table 2: Comparison between Our Algorithm and State-of-the-Art Placers

Benchmark	#Cell	Density	Total Disp. (sites)				Runtime (s)			
			[12]-Imp	[7]	[9]	Ours	[12]-Imp	[7]	[9]	Ours
des_perf_1	112644	90.58%	279545	474789	242622	188693	6.1	7.5	2.4	3.5
des_perf_a	108292	42.90%	81452	73057	72561	71044	2.5	3.8	2.3	1.8
des_perf_b	112644	49.71%	81540	72429	71888	70917	2.2	3.9	2.3	1.9
edit_dist_a	127419	45.54%	59814	60971	62961	56228	1.8	4.9	2.8	2.0
fft_1	32281	83.55%	54501	53389	46121	38821	1.0	1.3	0.7	0.9
fft_2	32281	49.97%	25697	21018	20979	20368	0.4	1.1	0.6	0.6
fft_a	30631	25.09%	19613	18150	18304	17375	0.2	1.2	0.6	0.5
fft_b	30631	28.19%	28461	21234	21671	20092	0.4	1.2	0.6	0.6
matrix_mult_1	155325	80.24%	80235	73682	71793	62026	4.0	5.4	3.6	3.3
matrix_mult_2	155325	79.03%	75810	65959	65876	58214	4.2	5.4	3.7	3.1
matrix_mult_a	149655	41.95%	46001	40736	40298	38013	1.6	5.7	3.4	2.6
matrix_mult_b	146442	30.90%	40059	37243	37215	35070	1.2	5.6	3.2	2.4
matrix_mult_c	146442	30.83%	42490	40942	40710	37907	1.4	5.6	3.2	2.4
pci_bridge32_a	29521	38.39%	27832	26674	26289	25917	0.3	1.2	0.6	0.4
pci_bridge32_b	28920	14.30%	27864	26160	26028	26081	0.2	1.0	0.4	0.4
superblue11_a	927074	42.92%	1786342	1983090	1742941	1595873	29.7	50.3	26.3	20.1
superblue12	1287037	44.72%	2015678	1995140	1963403	1716930	103.6	56.5	38.6	27.6
superblue14	612583	55.78%	1599810	1497490	1566966	1331144	16.7	48.1	17.7	13.1
superblue16_a	680869	47.85%	1173106	1147530	1135186	1055707	20.7	41.8	18.7	13.2
superblue19	506383	52.33%	806529	808164	781928	705239	10.5	29.6	13.2	10.7
Norm. Avg.			1.20	1.17	1.09	**1.00**	1.13	2.32	1.20	**1.00**



(a) Before



(b) After

Figure 6: Max displacement optimization.

Table 3: Results of Post-Processing

Benchmark	Avg. Disp.		Max. Disp.	
	Before	After	Before	After
des_perf_1	0.931	0.903	8.4	8.4
des_perf_a_md1	1.131	1.122	60.7	60.7
des_perf_a_md2	1.458	1.38	57.0	48.1
des_perf_b_md1	0.745	0.725	39.5	10.0
des_perf_b_md2	0.720	0.718	27.5	23.3
edit_dist_1_md1	0.762	0.752	5.7	5.7
edit_dist_a_md2	0.700	0.697	16.4	16.4
edit_dist_a_md3	0.839	0.837	31.4	31.4
fft_2_md2	0.916	0.905	9.6	7.1
fft_a_md2	0.637	0.631	34.3	34.3
fft_a_md3	0.611	0.605	11.3	11.3
pci_bridge32_a_md1	0.718	0.712	45.7	45.9
pci_bridge32_a_md2	0.876	0.872	18.1	18.1
pci_bridge32_b_md1	0.862	0.853	51.4	51.4
pci_bridge32_b_md2	0.791	0.785	61.7	61.7
pci_bridge32_b_md3	1.046	1.031	49.8	49.8
Norm. Avg.	1.01	**1.00**	1.23	**1.00**

fixed order optimization problem with a weighted sum of the maximum and average displacement as objective into another MCF problem for further optimization. Comparing with the champion of the IC/CAD 2017 Contest [16], we achived 18% less average displacement, 12% less maximum displacement, and much fewer routability-driven violations. We also compared with previous works and achieved a 9% improvement.

ACKNOWLEDGMENTS

This work is supported in part by The Research Grants Council of Hong Kong Special Administrative Region, China (Project No. CUHK14208914).

REFERENCES

[1] X. Xu, N. Shah, A. Evans, S. Sinha, B. Cline, and G. Yeric, "Standard cell library design and optimization methodology for ASAP7 PDK," in *Proc. ICCAD*, 2017, pp. 999–1004.
[2] L. Mattii, D. Milojevic, P. Debacker, Y. Sherazi, M. Berekovic, and P. Raghavan, "IR-drop aware design & technology co-optimization for N5 node with different device and cell height options," in *Proc. ICCAD*, 2017, pp. 89–94.
[3] S.-H. Baek, H.-Y. Kim, Y.-K. Lee, D.-Y. Jin, S.-C. Park, and J.-D. Cho, "Ultra-high density standard cell library using multi-height cell structure," in *Proc. SPIE*, vol. 7268, 2008.
[4] M. P.-H. Lin, C.-C. Hsu, and Y.-T. Chang, "Recent research in clock power saving with multi-bit flip-flops," in *Proc. MWSCAS*, 2011, pp. 1–4.
[5] D. D. Sherlekar, "Cell architecture for increasing transistor size," Jan. 14 2014, uS Patent 8,631,374.
[6] Y. Lin, B. Yu, and D. Z. Pan, "Detailed placement in advanced technology nodes: a survey," in *Proc. ICSICT*, 2016, pp. 836–839.
[7] 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 *Proc. ASPDAC*, 2017, pp. 450–455.
[8] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: fast legalization of standard cell circuits with minimal movement," in *Proc. ISPD*, 2008, pp. 47–53.
[9] J. Chen, Z. Zhu, W. Zhu, and Y.-W. Chang, "Toward optimal legalization for mixed-cell-height circuit designs," in *Proc. DAC*, 2017, pp. 52:1–52:6.
[10] B. Yu, X. Xu, J.-R. Gao, Y. Lin, Z. Li, C. Alpert, and D. Z. Pan, "Methodology for standard cell compliance and detailed placement for triple patterning lithography," *IEEE TCAD*, vol. 34, no. 5, pp. 726–739, May 2015.
[11] Y. Lin, B. Yu, Y. Zou, Z. Li, C. J. Alpert, and D. Z. Pan, "Stitch aware detailed placement for multiple e-beam lithography," *Integration, the VLSI Journal*, vol. 58, pp. 47–54, 2017.
[12] W.-K. Chow, C.-W. Pui, and E. F. Y. Young, "Legalization algorithm for multiple-row height standard cell design," in *Proc. DAC*, 2016, pp. 83:1–83:6.
[13] 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," *IEEE TCAD*, 2017.
[14] J. Vygen, "Algorithms for detailed placement of standard cells," in *Proc. DATE*, 1998, pp. 321–324.
[15] Y.-Y. Wu and Y.-W. Chang, "Mixed-cell-height detailed placement considering complex minimum-implant-area constraints," in *Proc. ICCAD*, 2017, pp. 65–72.
[16] N. K. Darav, I. S. Bustany, A. Kennings, and R. Mamidi, "ICCAD-2017 CAD contest in multi-deck standard cell legalization and benchmarks," in *Proc. ICCAD*, 2017, pp. 867–871.
[17] 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 *Proc. ISPD*, 2015, pp. 157–164.
[18] V. Yutsis, I. S. Bustany, D. Chinnery, J. R. Shinnerl, and W.-H. Liu, "ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement," in *Proc. ISPD*, 2014, pp. 161–168.
[19] N. K. Darav, A. Kennings, A. F. Tabrizi, D. Westwick, and L. Behjat, "Eh? placer: a high-performance modern technology-driven placer," *ACM TODAES*, vol. 21, no. 3, p. 37, 2016.
[20] Z. Király and P. Kovács, "Efficient implementations of minimum-cost flow algorithms," *arXiv preprint arXiv:1207.6381*, 2012.
[21] G. Chen, C.-W. Pui, W.-K. Chow, K.-C. Lam, J. Kuang, E. F. Y. Young, and B. Yu, "RippleFPGA: Routability-driven simultaneous packing and placement for modern FPGAs," *IEEE TCAD*, 2017.