Mixed-Cell-Height Standard Cell Placement Legalization

Chung-Yao Hung
National Tsing Hua University
Hsinchu, Taiwan
eruption@live.jp

Peng-Yi Chou
National Tsing Hua University
Hsinchu, Taiwan
super5516@gmail.com

Wai-Kei Mak
National Tsing Hua University
Hsinchu, Taiwan
wkmak@cs.nthu.edu.tw

ABSTRACT

A traditional standard cell library consists of various functional cells with the same height, which could speed up VLSI design flow since designers could align cells to placement sites in rows. However, in advanced nodes, cells are designed with different heights. With mixed cell heights, we can design simple cells (e.g. inverters) as single-row-height cells, and complex cells (e.g. flip-flops) as multiple-row-height cells. Nevertheless, placement legalization becomes more difficult with mixed-cell-height libraries. In this paper, we propose a parallel legalization method for mixed-cell-height standard cell libraries to minimize total displacement. Experimental results show that our method has a 19% improvement on average in displacement compared with the state-of-the-art work [9].

Keywords

Mixed-cell-height; Multiple-row-height; Legalization; Placement

1. INTRODUCTION

A traditional standard cell library consists of various functional cells with the same height[10], which could speed up VLSI design flow since designers could align cells to placement sites in rows. Thanks to the development of advanced technology node, we can achieve higher performance and smaller size with same functionality. As a result, the height of standard cell row has decreased steadily with reduced number of routing tracks. However, this can induce higher routing congestion within a cell[2], which makes it harder to fit complex circuits into such small height. Therefore, in order to increase layout efficiency, modern standard cell library adopts the concept of mixed-cell-height[5], i.e., a library containing not only single-row-height cells but also multiple-row-height cells. With mixed cell heights, we can design simple cells (e.g. inverters) as single-row-height cells, and complex cells (e.g. flip-flops) as multiple-row-height cells.

GLSVLSI '17, May 10-12, 2017, Banff, AB, Canada

DOI: http://dx.doi.org/10.1145/3060403.3060473

Nevertheless, some VLSI design stage, namely the placement stage, will be influenced by using mixed-cell-height libraries. The placement stage could be divided into three steps: (i) global placement, (ii) legalization, (iii) detailed placement. During global placement, each cell will be placed to optimize some characteristics of the layout, e.g. minimize total wirelength, balance cell density. Because global placement allows cells to overlap with each other, we need to resolve cell-overlapping problems and to align every cells to placement sites by moving cells, that is what legalization does. Finally, detailed placement will further improve the quality of legalized solution by some techniques, such as shifting or swapping cells. As moving a multiple-row-height cell will have an impact on cells on multiple rows, it makes legalization and detailed placement more difficult to accomplish.

There are only a few previous works for mixed-cell-height standard cell design placement. Wu et al. [11] are the first to handle legalization and detailed placement with double-row-height cells. By cell pairing and cell expansion, they transform all single-row-height cells into double-row-height cells. Thus, a traditional detailed placer could still be adopted. However, they did not take power-rail alignment constraints into consideration, which make their result impractical. Besides, their method could only deal with cells up to double-row-height. Chow et al. [4] proposed the first placement legalization algorithm for multiple-row-height cell designs to minimize displacement. They greedily find a legalization solution with minimal displacement for each cell within the defined local region around its global placement position in a sequential manner. They also implemented an integer linear programming (ILP) based algorithm for comparison. Lin et al. [7] proposed a two-stage method composed of legalization stage and detailed placement stage for multiple-row-height cell designs. The chain move method they proposed will iteratively pick the largest unmoved cell at the time and try to place it in the defined local region, and handle the overlapped cells sequentially. Combining chain move with the legalization approach in [4], they could find an overlap-free solution in the legalization stage. Wang et al. [9] modified Abacus[8], a well-known legalization method for single-row-height cell designs, to handle multiple-row-height cell designs. For each cell, they will evaluate and find the best position by trying to place it in each row. Therefore, they could get a better solution than [4]. The above works [4], [7], and [9] have one thing in common that they do the legalization cell by cell, whereas we legalize multiple cells simultaneously.

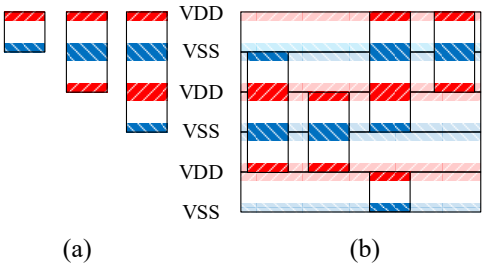

Figure 1: (a) Standard cells with multiple-row-height. (b) Some examples of power-rail alignment with multiple-row-height cells.

The rest of this paper is organized as follows. In Section 2, we introduce the difficulties faced in legalization using mixed-cell-height libraries, and give the problem formulation of this work. Section 3 describes the proposed method. Experimental results are shown in Section 4. Finally, Section 5 gives the conclusion.

2. PRELIMINARIES

2.1 Legalization

Traditionally, for a legal placement, it has to satisfy the following constraints:

- Cells must be aligned to placement sites.
- Cells are overlap-free.

However, as mentioned in Section 1, global placement will try to optimize some characteristics of the layout. Thus, during legalization, we will try not to move each cell far from its global placement position to preserve the characteristics. In other words, the cell displacement should be minimized.

2.2 Power-Rail Alignment Constraint

In reality, standard cells have to connect to VDD on the top or bottom side, and to VSS on the other side. Therefore, when it comes to multiple-row-height cells, legalization is not as simple as it used to be. For multiple-row-height cells with even row height, both sides of the cell are either connected to VDD or VSS, which means they could only be placed on rows with proper power-rail alignment. On the other hand, cells with odd row height could be placed at any row by vertically flipping them if necessary. Figure 1 shows some examples.

2.3 Problem Formulation

Given a global placement result containing multiple-row-height cells, the objective is to legalize the placement while minimizing total cell displacements. The total cell displacements is obtained through the following equation:

$$\sum_{i=1}^{n}(|x_i - x_i'| + |y_i - y_i'|)$$

where n denotes the total number of cells, x_i/y_i the x-/y-coordinate of the final position of cell i, x_i'/y_i' the x-/y-coordinate of the global placement position of cell i.

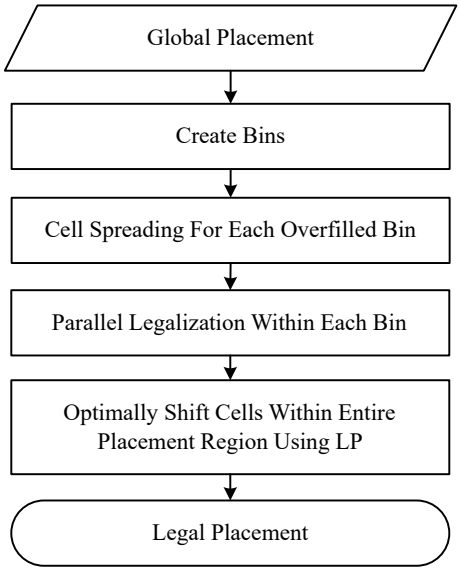

Figure 2: The overall flow of the proposed method.

3. PROPOSED ALGORITHM

3.1 Overall Flow

Figure 2 gives the overall flow of the proposed method. First, we partition the placement region to form uniform bins, and generate an initial cell-to-bin assignment according to the global placement solution. Then, we spread the cells from high density bins to neighboring and relatively low density bins. After that, a legalization solution is obtained through an integer linear programming based formulation. Finally, a linear programming based approach is performed to improve the solution quality by optimally shifting all the cells horizontally within the entire placement region. We will describe the first step here, and the rest of the steps will be detailed in the following sub-sections.

The purpose of creating bins is to speed up the ILP formulation in the following step. Although ILP formulation guarantees to obtain an optimal solution under the given objective and constraints, it is time consuming when the number of variables is large. Hence, we divide the entire layout into bins and solve them as sub-problems. However, the act of dividing will harm the optimality of the solution. Hence, we also propose some techniques (described in Section 3.3) to mitigate the effect.

After creating bins, we generate an initial cell-to-bin assignment that assigns cells to proper bins according to their global placement positions. As a result, bins in dense areas will be assigned too many cells and make them impossible to have a feasible placement solution. We call those bins overfilled bins. In order to find a feasible placement solution, we need to spread some cells in overfilled bins to neighboring bins that have enough space to accommodate them.

3.2 Cell Spreading

During cell spreading, we want to determine the cells we should move out from overfilled bins and the bins we should move them to. At the same time, we want to minimize

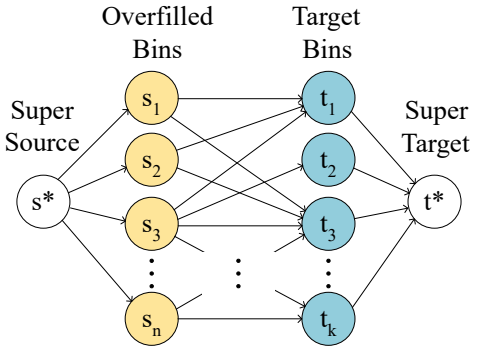

Figure 3: The structure of our network flow model.

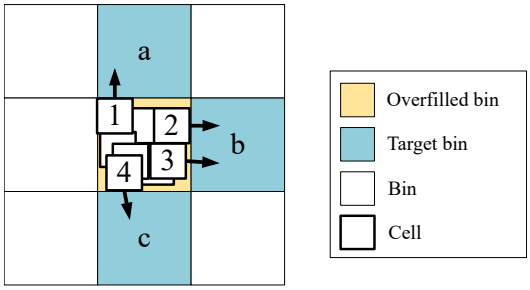

Figure 4: An example of *boundary cell moving*. Since cells that are nearer to the target bin will be moved with higher priority, cell 1 will be moved to bin a, cell 2 and 3 will be moved to bin b, and cell 4 will be moved to bin c.

the total displacement. Therefore, we define a cell's moving distance as follows:

Bin-wise displacement: The bin-wise displacement of a cell is the Manhattan distance between the bins to which the cell is assigned before and after movement.

And then we could formulate this problem as follows:

The Cell Spreading Problem: Given a cell-to-bin assignment solution, for each overfilled bin, determine the cells to move out and the target bins to which they should be moved to, while minimizing the total bin-wise displacement.

We further divide the cell spreading problem into two parts: (i) *budgeting* and (ii) *boundary cell moving*, and solve them sequentially. For *budgeting*, we use network flow model[1] to determine how much cell area should be moved out from overfilled bins to nearby bins that have extra spaces. We call those nearby bins target bins. For each overfilled bin s_i, we only assign a few nearby bins that are within a range of bin-wise distance as its target bins t_j. The structure of our network flow model is shown in Figure 3. There are four columns of nodes, super source, overfilled bins, target bins, and super target, respectively. Each overfilled bin node s_i has an edge from the super source node s^*, and each target bin node t_j has an edge to the super target node t^*. Each overfilled bin node s_i has an edge to target bin node t_j if that bin is a target bin of this overfilled bin. In addition, the capacity and cost of an edge between the first two columns are set to be the amount of area needed to be moved out from the overfilled bin and 0, respectively. The capacity and cost of an edge between the middle two columns are set to be infinity and the corresponding bin-wise displacement, respectively. The capacity and cost of an edge between the last two columns are set to be the amount of free space in the corresponding target bin and 0, respectively.

Since we know the budgeted amount of cell area to move from an overfilled bin s_i to a target bin t_j after *budgeting*, we then move cells in s_i that are nearest to t_j to the target bin t_j according to the budgeted amount, as shown in Figure 4. This completes the *boundary cell moving* part.

3.3 Parallel Legalization

In the previous subsection, we have determined which cell should be placed in which bin. Therefore, we still need to place each cell onto a proper placement site within each bin. To legalize the placement while minimizing total cell displacement, we first enumerate a number of valid candidate positions of each cell within the bin it belongs to. To

Table 1: Notations of the ILP Formulation

C	Set of cells
R	Set of rows
Q	Set of columns
K_c	Set of candidate positions of cell c
S_{rq}	Set of candidate positions occupying site (r, q)
D_c^k	Displacement of k-th candidate position of cell c from c's global placement position
p_c^k	0/1 variable, $p_c^k = 1$ if k-th candidate position of cell c is chosen

be more specific, we enumerate candidate positions within a fixed range around each cell's global placement position. And then compute the cell placement inside each bin using an ILP formulation inspired by [6]. Note that for each cell that has been moved during cell spreading stage, we will enumerate candidate positions within a fixed range from the bin boundary nearest its global placement position.

A valid candidate position is a position that when cell is placed on it, there will be no power-rail alignment violation and overlapping with macros, as shown in Figure 5. Due to the power-rail alignment constraint of multiple-row-height cells, we allow them to be placed partially outside the top or bottom bin boundary. Therefore, for a multiple-row-height

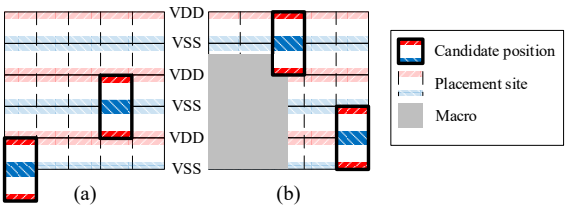

Figure 5: (a) Some examples of valid candidate positions that satisfy power-rail alignment and do not overlap with macros nor obstacles. (b) Some examples of invalid candidate positions that have power-rail alignment violation or overlap with macros/obstacles.

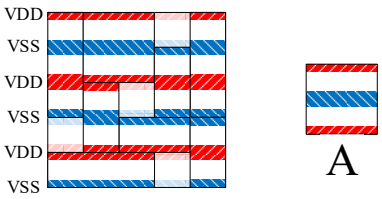

Figure 6: Cell A is left unplaced even though the total dead spaces is larger than A's area.

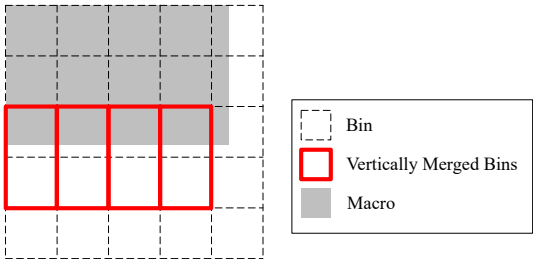

Figure 7: An example of vertically merging bins. After merging, bins with little free placement space could find solution more easily.

cell, we will generate a candidate position even if it crosses the top or bottom boundary of the bin that it has been assigned to, as long as it is a valid position for the cell. Note that we can solve the ILPs for all bins in the same row in parallel since their placement problems are independent, but since we allow a multiple-row-height cell to protrude to a vertically adjacent bin, we process the bins row by row from bottom to top. Thus, if any candidate position is chosen during the ILP process, the cell will be treated as an obstacle from other bin's viewpoint afterwards.

The notations used in our ILP formulation are listed in Table 1. Having the notations, the ILP formulation is given as follows:

minimize

$$\sum_{c \in C} \sum_{k \in K_c} D_c^k \cdot p_c^k$$

subject to

$$\sum_{k \in K_c} p_c^k = 1, \forall c \in C \tag{1}$$

$$\sum_{p_c^k \in S_{rq}} p_c^k \leq 1, \forall r \in R, q \in Q \tag{2}$$

In the formulation, constraint (1) ensures that every cell will be placed and that for each cell, only one candidate position will be chosen. Constraint (2) prevents cells from overlapping with each other.

However, even if the total area of cells assigned within the bin is less than the available bin area, we might not find a legal placement due to dead spaces as shown in Figure 6. Hence, after each run of ILP process, we will mark those bins that could not find feasible solution as *infeasible bins*, and merge them with a horizontal neighboring bin to form bin clusters. By doing so, every cell within a cluster will have more candidate positions, then the solution space could be enlarged.

Additionally, when there are large macros in the layout, we will inevitably have some bins that have much of their space occupied by macros. Chances are that these bins cannot find feasible solutions easily even after they are merged with their horizontal neighbors, since their neighbors also have little free placement space. Besides, the solution quality will be degraded because even if there is a feasible placement after merging a few consecutive horizontal bins, some cells may be moved far away and results in higher displacement. To prevent this from happening, we simply merge each bin partially occupied by macros with its vertical neighboring bin before candidate position enumeration, as shown in Figure 7.

Table 2: Notations of the LP Formulation

C	Set of cells
C_r	Set of cells on row r with increasing order of x-coordinate
R	Set of rows
W_c	The width of cell c
X_c	The x-coordinate of the global placement position of cell c
x_c	The x-coordinate of the final position of cell c
x_{c+1}	The x-coordinate of the final position of cell c+1, which is to the right of cell c
d_{x_c}	Horizontal cell displacement of cell c from its global placement position

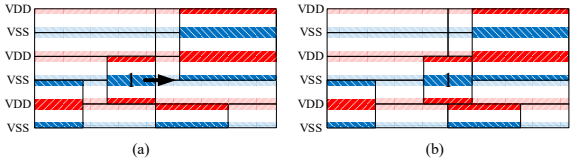

Figure 8: The quality of a legal placement solution may be further improved by shifting cells horizontally across bin boundary. If the global placement position of cell 1 is within the bin at right, we could horizontally shift it to reduce cell displacement.

3.4 LP-based Refinement

As mentioned before, bin boundaries will restrict solution space, as shown in Figure 8, so we want to further improve the solution quality by doing post-processing. After obtaining a legal placement solution, we optimally shift the cells horizontally within the same row while keeping their relative order. We use a linear programming (LP) formulation to accomplish this task. The notations used in our LP formulation are listed in Table 2. Having the notations, the LP formulation is given as follows:

minimize

$$\sum_{c \in C} d_{x_c}$$

subject to

$$x_c + W_c \leq x_{c+1}, \forall c \in C_r, \forall r \in R \tag{3}$$

Table 3: Experimental Results Compared with [9] and [4].

Benchmark	#S. cells	#D. cells	Density	Average Displacement(sites)				Runtime(s)			
				Ours	MLL[4]	ILP[4]	[9]	Ours	MLL[4]	ILP[4]	[9]
des_perf_1	103842	8802	0.91	1.44	3.32	2.13	3.46	1162	7.0	4099	18.0
des_perf_a	99775	8513	0.43	0.65	0.96	0.66	0.68	27	2.6	194	6.1
des_perf_b	103842	8802	0.5	0.62	0.85	0.62	0.64	23	2.4	251	6.2
edit_dist_a	121913	5500	0.46	0.44	0.47	0.45	0.47	76	1.9	206	7.8
fft_1	30297	1984	0.84	1.06	1.81	1.58	1.55	215	1.1	777	1.3
fft_2	30297	1984	0.5	0.63	0.86	0.66	0.64	12	0.4	73	0.8
fft_a	28718	1907	0.25	0.56	0.64	0.60	0.64	17	0.3	38	0.8
fft_b	28718	1907	0.28	0.62	0.80	0.73	0.62	17	0.4	62	1.0
matrix_mult_1	152427	2898	0.8	0.38	0.53	0.49	0.48	148	3.9	967	9.1
matrix_mult_2	152427	2898	0.79	0.36	0.49	0.45	0.44	132	4.0	825	8.9
matrix_mult_a	146837	2813	0.42	0.25	0.33	0.27	0.27	55	1.6	151	9.3
matrix_mult_b	143695	2740	0.31	0.23	0.30	0.25	0.25	49	1.3	128	8.9
matrix_mult_c	143695	2740	0.31	0.27	0.29	0.27	0.27	58	1.4	139	9.0
pci_bridge32_a	26268	3249	0.38	0.86	0.95	0.88	0.88	10	0.3	49	0.8
pci_bridge32_b	25734	3180	0.14	0.90	0.96	0.95	0.52	9	0.2	15	0.7
superblue11_a	861314	64302	0.43	1.85	1.96	1.85	1.86	1798	23.4	3074	80.2
superblue12	1172586	114362	0.45	1.33	1.63	1.45	1.63	1487	106.5	5079	91.1
superblue14	564769	47474	0.56	2.18	2.62	2.56	2.38	3436	17.1	3360	70.3
superblue16_a	625419	55031	0.48	1.61	1.73	1.61	1.68	965	21.7	2471	61.0
superblue19	478109	27988	0.52	1.48	1.60	1.52	1.68	929	10.9	1849	44.6
Norm.				1.00	1.30	1.13	1.19				

$$d_{x_c} \geq x_c - X_c, \forall c \in C \tag{4}$$

$$d_{x_c} \geq -(x_c - X_C), \forall c \in C \tag{5}$$

In the formulation, constraint (3) ensures the same cell ordering is kept and prevents cells from overlapping with each other. Since the final position of cell c may be to the right or to the left of c's global placement position, we use constraints (4) and (5) to get a tight lower bound on the magnitude of c's horizontal displacement d_{x_c}. Since d_{x_c} appears in the minimization objective of the LP, it will force d_{x_c} to be exactly equal to the horizontal displacement of c.

4. EXPERIMENTAL RESULTS

We implemented the proposed method using C++ programming language, and performed experiments on a Linux workstation with Intel Xeon 2.6GHz CPU with 28 threads and 64GB memory. Gurobi is selected as the LP and ILP solver. Benchmarks used in the experiments are from the authors of [4]. They modified the benchmarks from the 2015 ISPD Detailed-Routing-Driven Placement Contest[3] by randomly picking 10% of the cell to double their heights and half their widths. To strike a balance between better solution quality and faster runtime, we set the bin size as 160 placement sites wide and 9 rows high. In addition, for the ILP formulated for each bin, we would terminate early if no better solution can be found for a period of 15 seconds. During cell spreading stage, bins with density higher than 0.9 are treated as overfilled bins.

Table 3 shows the statistics of the benchmarks and the legalization results. "#S. cells" gives the total number of single-row-height cells, "#D. cells" the total number of double-row-height cells, "Density" the density of design, "Average Displacement(sites)" the average displacement measured in the number of the placement site width, "Runtime(s)" the running time in seconds. Finally, "Ours," "MLL[4]," "ILP[4]," and "[9]" give the results from our method, the fast greedy legalizer proposed in [4], the ILP implemented by the authors in [4], and the method proposed in [9], respectively[1].

We first compare our results with [4]. They proposed a fast greedy legalizer and implemented an ILP based algorithm for comparison. However, judging from their results, the only advantage of their greedy legalizer is the runtime. For fair comparison, we will focus on the effectiveness and efficiency of their ILP based algorithm. We could see from Table 3 that our method could achieve better average displacement with as much as 32% improvement on benchmark des_pref_1 and 13% improvement on average.

Next, we compare our results with the latest work on legalization for mixed-cell-height standard cells[9]. In addition to displacement, they also put efforts on minimizing the half perimeter wirelength (HPWL) and get better results than [4] did. But since our objective does not include HPWL, we only compare the results of displacement with them. We could see from Table 3 that we get better quality in all but one case with a 19% improvement on average.

5. CONCLUSION

For circuit designs in advanced technologies, mixed-cell-height standard cell libraries become popular. In this paper we proposed a parallel legalization method which uses ILP formulation to minimize total cell displacement. We can see from the experimental results that our method has a 19% improvement on average in displacement compared with the state-of-the-art work [9].

Acknowedgements

This work was supported in part by the Ministry of Science and Technology under grant MOST 104-2628-E-007-003-MY3.

[1]The results for [4] are directly taken from their paper and their platform was Intel Xeon 3.4GHz with 32GB memory, *lpsolve* was chosen as the ILP solver. The results for [9] are directly taken from their paper and their platform was Intel Xeon 2.93GHz with 48GB memory.

6. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] 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*, volume 7268, pages 72680C–72680C–8, 2008.

[3] 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 the 2015 Symposium on International Symposium on Physical Design*, ISPD '15, pages 157–164, 2015.

[4] W.-K. Chow, C.-W. Pui, and E. F. Y. Young. Legalization algorithm for multiple-row height standard cell design. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 83:1–83:6, 2016.

[5] S. Dobre, A. B. Kahng, and J. Li. Mixed cell-height implementation for improved design quality in advanced nodes. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '15, pages 854–860, 2015.

[6] S. Li and C.-K. Koh. Mixed integer programming models for detailed placement. In *Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design*, ISPD '12, pages 87–94, 2012.

[7] 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 the 35th International Conference on Computer-Aided Design*, ICCAD '16, pages 7:1–7:8, 2016.

[8] P. Spindler, U. Schlichtmann, and F. M. Johannes. Abacus: Fast legalization of standard cell circuits with minimal movement. In *Proceedings of the 2008 International Symposium on Physical Design*, ISPD '08, pages 47–53, 2008.

[9] 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 the 22nd Asia and South Pacific Design Automation Conference*, ASP-DAC '17, 2017.

[10] J. Wang, A. K. Wong, and E. Y. Lam. Standard cell layout with regular contact placement. *IEEE Transactions on Semiconductor Manufacturing*, 17(3):375–383, Aug 2004.

[11] 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, Sept 2016.