A Mixed-Height Standard Cell Placement Flow for Digital Circuit Blocks*

Yi-Cheng Zhao[1], Yu-Chieh Lin[1], Ting-Chi Wang[1],
Ting-Hsiung Wang[2], Yun-Ru Wu[2], Hsin-Chang Lin[2], Shu-Yi Kao[2]
[1]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[2]Realtek Semiconductor Corp., Hsinchu, Taiwan

***Abstract*—In this paper, we present a mixed-height standard cell placement flow for digital circuit blocks. To our best knowledge, commercial tools currently do not support this type of flow in a fully automated manner. In our placement flow, we leverage a commercial placement tool and integrate it with several new point tools. Promising experimental results are reported to demonstrate the efficacy of our placement flow.**

I. Introduction

Standard cell based design methodology has been widely used to speed up circuit design. A typical standard cell library is a collection of various equal-height functional cells. For several decades, most research efforts in placement have been focused on standard cells of the same height [1]. The placement region is divided into rows, and each row shares the same height with standard cells. Cells must be placed into rows without any overlapping.

A standard cell of larger height provides better performance but inversely has larger area and consumes more power than one with smaller height. As a result, a smart strategy for designing a digital circuit block should try to mix the usage of cells with different heights for achieving better design quality. In this paper, we study a mixed cell-height placement problem. In our problem, standard cells come from two distinct cell-height standard cell libraries in the same technology node. Current commercial tools do not support mixing cells with different heights during the placement stage of a digital circuit block.

A prior work in [2] proposed a placement approach for mixed-height standard cells. In order to legalize the placement solution (i.e., cells of equal height must be placed in regions with the same row height), two operations (displacement of cells, and swapping of cells across different libraries) were adopted. The authors of [2] pointed out that cell swapping across different libraries was required much more frequently than cell displacement, which could change the netlist significantly.

On the premise of no cell swapping across different libraries, we propose in this paper a mixed cell-height placement flow for digital circuit blocks. To make the proposed flow feasible, we leverage a commercial placement tool (i.e., Cadence Innovus [3]) and integrate it with several new point tools to automatically produce a mixed cell-height placement from a gate-level netlist. We empirically show that mixed cell-height placement can meet timing constraint through our proposed flow for all test cases.

II. Problem Formulation and Overview of Our Placement Flow

In this section, we first introduce the problem formulation for mixed cell-height placement and then give an overview of our placement flow.

*This work was supported in part by Realtek Semiconductor Corp.

A. Problem Formulation

A gate-level netlist (consisting of standard cells with two different heights, e.g., 9-track and 12-track), two cell libraries, a timing constraint, and floorplan constraints (i.e., block utilization and aspect ratio) are given as inputs. The addressed problem asks to produce a legal cell placement, where the placement region is determined according to the total cell area and the floorplan constraints, and each cell is positioned in a row of the same height without any cell overlapping. The objective of the problem is to meet the timing constraint and to minimize the half perimeter wirelength.

In our problem, the following layout constraints are imposed as well.

- C_1: To honor the well-to-well spacing rule, two cells of different heights must be separated by at least a specific horizontal distance.
- C_2: To avoid P/G rail encroachment, a minimum vertical distance is also in need between two cells of different heights.

Note that the set of cells in the gate-level netlist can be separated into two disjoint subsets according to their heights. The subset has smaller total cell area is called *target set*, and each cell in the target set is called *target cell*. Without loss of generality, we assume cells with the larger height are target cells in the rest of this paper.

B. Overview of Our Placement Flow

Our mixed cell-height placement flow is shown in Fig.1. It consists of four main stages: single cell-height placement, region creation, mixed cell-height placement, and region update. The flow is explained in the next paragraph, while the details of each main stage are presented in Sections III-VI.

The placement flow starts with the single cell-height placement stage, which first unifies all cells to have the same height by converting target cells into ones with the smaller cell height, and then produces a single cell-height placement that tries to place target cells closer so as to facilitate the creations of regions. Next it enters the region creation stage, which first clusters all target cells into disjoint groups based on the single cell-height placement, and then determines a region for each group of target cells. Then the mixed cell-height placement stage is executed to restore the shape of each target cell and to place cells such that each target cell is placed in its corresponding region while each non-target cell is placed outside the regions; at the end of this stage, a mixed cell-height placement is produced and its legality is checked. If the placement is legal (i.e., no cell overlapping), we output the result. Otherwise, the region update stage is invoked. Each illegal region (i.e., a region which is not large enough and leads to cell overlapping) is enlarged in the region update stage, and then the mixed cell-height placement stage is performed again. The iterative process between the mixed cell-height placement stage and region update stage terminates when a legal placement is obtained or an iteration limit is reached.

978-3-9819263-2-3/DATE19/©2019 EDAA

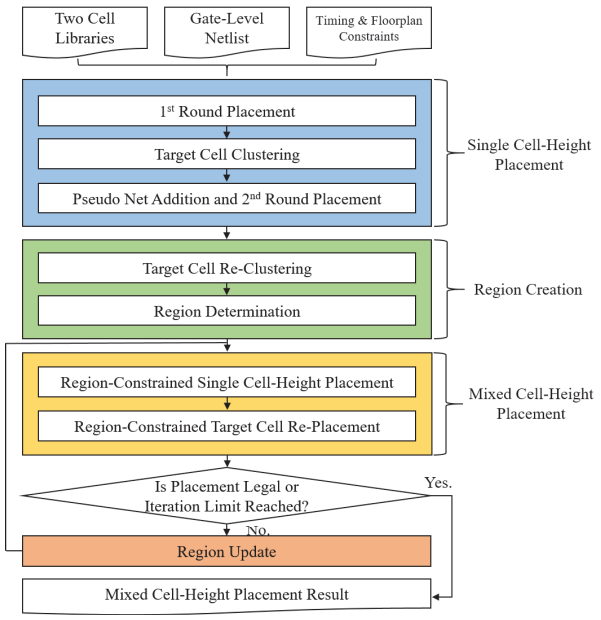

Fig. 1. The proposed placement flow.

III. Single Cell-Height Placement

The aim of the single cell-height placement stage is to obtain a single cell-height placement that will be utilized to guide the region creation for target cells in the next stage. This stage includes three steps: (1) 1st round placement, (2) target cell clustering, and (3) pseudo net addition and 2nd round placement.

A. 1st Round Placement

In this step, we use a commercial tool (i.e., Cadence Innovus [3]) to place cells and obtain the 1st round placement. The placement region is determined according to the total cell area as well as the block utilization and aspect ratio. Since the commercial placement tool cannot directly place cells with two different heights, we unify all cells to have the same height by converting each target cell into one with the smaller height while maintaining the same or similar cell area. From now on, a target cell with shape conversion is called *modified target cell*. The shape conversion enables the commercial placement tool to produce a single cell-height placement, i.e., the 1st round placement, under the given timing constraint.

B. Target Cell Clustering

To help reduce the area overhead induced by the minimum spacing between two adjacently placed cells of different heights due to the layout constraint C_1 or C_2, in the target cell clustering step, we aim at identifying modified target cells that are close to each other in the 1st round placement and cluster them to form a group.

1) Graph Construction: We first construct an edge-weighted undirected graph G to model the distance relationship between each pair of modified target cells. In G, each vertex represents a modified target cell, and if the distance between two modified target cells is less than a user-defined threshold d (whose unit is a placement site width), an edge connecting two corresponding vertices is constructed. In addition, each edge (u, v) is associated with a positive weight which is set to be $d - dis(u, v)$, where $dis(u, v)$ is the distance between the two modified target cells corresponding to vertices u and v. A larger (smaller, respectively) edge weight implies a higher (lower, respectively) chance for the two corresponding modified target cells to be grouped together. Finally, for each vertex i, a self-loop edge (i, i) is also added to G, and the edge weight is set to be the average weight of all edges incident with i. In our current implementation, we set d to be a number between 137 and 525, depending on the size of the design.

2) Markov Cluster Algorithm: Markov cluster algorithm (MCA) is a powerful and universal clustering approach. MCA uses the concept of flow simulation on a graph to make nature grouping without the need to specify the amount of groups to be produced in advance. Detailed mathematical concepts supporting MCA can be found in, e.g., [4].

Suppose there are n vertices in G, and they are numbered by 1, 2,..., n. We use an adjacency matrix M to represent G, where m_{ij} denotes the element at row i and column j of M and is set to be the weight of edge (i, j). If there is no edge connecting vertices i and j, m_{ij} and m_{ji} are both set to be 0.

MCA simulates random walk based on M to make nature grouping. Algorithm 1 shows how MCA works on M to cluster the set of modified target cells into disjoint groups. It first normalizes M (line 1); i.e., each element m_{ij} is divided by the sum of all the elements in column j so as to make m_{ij} represent the transition probability from vertex i to vertex j. It then enters an iterative process (lines 2-9). At each iteration, it first performs matrix multiplication, i.e., $M \times M$, for e times to perform random walk on each vertex (lines 4-6). Then each element of M is raised to the r-th power (line 7) for changing the distribution of transition probability such that vertices preferred to be grouped together are further favored and less popular vertices are demoted. Before the current iteration ends, M is normalized again (line 8). The iterative process terminates when the matrices produced by two consecutive iterations are approximately equal (line 9). Finally the clustering result is obtained according to all non-zero elements of each row of M (line 10). For each row i, if m_{ij} is non-zero, then vertices i and j are in the same group. Note that each cell only belongs to one specific group. MCA uses two parameters e and r (lines 4 and 7) to control the tradeoff between solution quality and runtime. In our implementation, we set e and r to be 29 and 2, respectively.

Algorithm 1 Markov Cluster Algorithm (MCA)

```
1:  M ← Normalize(M)
2:  repeat
3:      M_pre ← M
4:      for i = 0; i < e; i++ do
5:          M ← M * M
6:      end for
7:      M ← Power(M, r)
8:      M ← Normalize(M)
9:  until M ≈ M_pre
10: Produce the clustering solution according to all non-zero
    elements of M
```

3) Pre-processing: We empirically observed that the runtime of MCA could become unacceptable when it runs on large graphs. To cope with this problem, for any design with 5,000 modified target cells or more, we pre-process the set of all modified target cells by partitioning it into a collection of disjoint subsets each of which has a manageable size, and then apply MCA on each subset.

The pre-processing is done by a method called Jarvis-Patrick clustering (JPC) [5]. There are two user-defined parameters k and k_{min} in JPC, where k is less than or equal to the number of all modified target cells, and $k_{min} \leq k$. For each modified target cell i, let $N(i,k)$ denote the set of k nearest modified target cells to i (measured by the distance

between cells), including i itself. JPC is able to divide the set of all modified target cells into disjoint subsets such that the following property holds for each subset S: For each modified target cell i in S, there exists another modified target cell j in S such that (1) $N(i,k)$ and $N(j,k)$ have at least k_{min} cells in common, i.e., $|N(i,k) \cap N(j,k)| \geq k_{min}$, and (2) $i \in N(j,k)$ or $j \in N(i,k)$. In our experiments, we set k to be between 60 and 80, and set k_{min} to be 2.

C. Pseudo Net Addition and 2nd Round Placement

In this step, we create a pseudo net to connect all modified target cells in each group. Meanwhile, an appropriate weight is assigned to each pseudo net so as to encourage the commercial placement tool to consider these pseudo nets with a higher priority. We run the commercial placement tool in a timing-driven incremental placement manner to produce the 2nd round placement. Owing to the fact that each group of modified target cells is additionally connected by a pseudo net, the commercial placement tool is able to place such target cells even closer.

IV. Region Creation

The region creation stage aims at re-forming the groups of modified target cells based on the result of the single cell-height placement stage and determining the region for each new group. These regions will be used in the next stage for placing target cells with their shapes restored. The region creation stage includes two steps: (1) target cell re-clustering, and (2) region determination.

A. Target Cell Re-Clustering

Due to the position of each cell has been changed after 2nd round placement, we re-cluster all modified target cells again and get the new groups. This step is done by using the same clustering approach given in Section III-B

B. Region Determination

After re-clustering all modified target cells, we create in this step a placement region for each group of target cells. Each region specifies that only the cells in the corresponding group can be placed inside, and no other cells are allowed to be placed in it.

For each cell group, we traverse all its cells to get its total cell area as well as the minimum bounding box enclosing all its cells. Then, we calculate the required area for each cell group according to its total cell area and the block utilization. Finally, we re-size the bounding box for each cell group to get the corresponding region by considering the required area, the original height of a target cell, and the placement site width.

After the region for each cell group is created, we continue to check if there exists region overlapping. If found, we keep merging two overlapping regions into one larger region till no more overlapping exists. To merge two overlapping regions into one region, we first get the minimum bounding box enclosing the two regions, and then set the sum of the two region areas to be the required region area after merging. At last, we re-size the bounding box to get the corresponding region by considering the required region area, the original height of a target cell, and the placement site width.

Note that when a new region is created, a minimum movement might be necessary for it to be aligned with the overall metal and poly tracks of the block.

V. Mixed Cell-Height Placement

The purpose of the mixed cell-height placement stage is to take advantage of the regions formed in the previous stage and produce a mixed cell-height placement such that each target cell has its shape restored and placed in the corresponding region while each non-target cell is placed outside any region.

A natural way to obtain a mixed cell-height placement is to restore the shape of each target cell and apply the commercial placement tool to perform region-constrained timing-driven placement such that each target cell is placed in the corresponding region while each non-target cell is placed outside any region. To this end, we need to restore the shape of each target cell, and provide the commercial placement tool with *region constraints* that force the tool to place each group of target cells into the corresponding region. Besides, we need to create two types of cell rows in the whole placement area. The cell rows in each region have their heights equal to the original height of a target cell, while the cell rows outside any region have their heights equal to the height of a non-target cell. Finally, *blockages* are added along the four boundaries of each region to account for the required minimum spacing between cells of different heights. All cells are forbidden to be placed in any blockage area.

An example of mixed cell-height placement of design AES produced by the above-mentioned approach is shown in Fig. 2(b), in which non-target cells are in gray, and regions containing target cells (with their shape recovered) are in other colors. Unfortunately some misbehaviors are observed from Fig. 2(b). For example, a huge amount of area marked by the left green dash circle has no cells placed there, while a part of the area marked by the right green dash circle is crowded with non-target cells. Besides, this placement violates the timing constraint. To fix such misbehaviors, our mixed cell-height placement stage uses a two-step approach, including (1) region-constrained single cell-height placement, and (2) region-constrained target cell re-replacement.

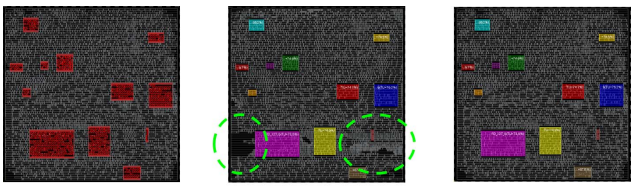

Fig. 2. Various region-constrained placement results of design AES. (a) Region-constrained single cell-height placement. (b) Poor mixed cell-height placement. (c) Good mixed cell-height placement.

A. Region-Constrained Single Cell-Height Placement

In this step, we apply the commercial placement tool to perform region-constrained timing-driven placement such that each target cell is placed in the corresponding region without restoring its shape while each non-target cell is placed outside any region. The result is a region-constrained single cell-height placement, and an example is shown in Fig. 2(a), in which target cells are in red, and non-target cells are in gray. For this placement, timing constraint is met, but the shape of each target cell is not restored yet.

B. Region-Constrained Target Cell Re-placement

By comparing Fig. 2 (a) and 2 (b), we observe that the region-constrained single cell-height placement in Fig. 2 (a) has a better placement for non-target cells than the one in Fig. 2 (b). Therefore, in this step, the position of each non-target cell is inherited from the region-constrained single cell-height placement and is kept intact. To produce a mixed cell-height placement, we restore the shape of each target cell and force the commercial placement tool to re-place all target cells within their corresponding regions subject to the timing constraint.

Fig. 2 (c) shows an example of mixed cell-height placement of design AES produced by our two-step approach. This result is better than the one in Fig. 2(b) as it meets the timing constraint.

TABLE I
STATISTICS AND EXPERIMENTAL RESULTS OF TEST CASES

Test Case (Clock Period)	Flow	#Cells 9-track/12-track	#Nets	WL (μm)	Block Area (μm^2)	WNS (ps)	TNS (ps)
AES (800ps)	9-track	6044/0	6169	59633	9450	129	0
	12-track	0/5834	5966	70977	13547	144	0
	mixed	5446/587	6157	62324	9552	109	0
AES (700ps)	9-track	6058/0	6198	61624	10041	86	0
	12-track	0/5977	6108	68000	12671	70	0
	mixed	5109/844	6079	69980	10365	75	0
AES (600ps)	9-track	6103/0	6253	64447	11560	-28	-496
	12-track	0/5993	6122	72593	14043	-1	-1
	mixed	4881/1164	6177	74750	11808	38	0
ECG (900ps)	9-track	54843/0	55390	442090	60934	-148	-14490
	12-track	0/48716	49266	481100	78899	118	0
	mixed	52984/1173	54704	489520	61365	262	0
ECG (800ps)	9-track	55518/0	56066	402100	63207	-109	-138
	12-track	0/48656	49206	426000	78899	32	0
	mixed	52483/2218	55204	461800	64597	210	0
ECG (700ps)	9-track	55661/0	56210	491940	67948	-73	-202
	12-track	0/49187	49738	478820	80683	-63	-120
	mixed	50566/3845	54959	510160	68402	71	0
TBPC (1000ps)	9-track	188512/0	189307	1837100	240688	-494	-366790
	12-track	0/180397	181197	2490700	324261	-132	-12391
	mixed	184235/1922	186955	1908600	242467	89	0
TBPC (900ps)	9-track	190457/0	191258	1886000	246462	-276	-161498
	12-track	0/181490	182290	1995300	325664	-35	-206
	mixed	182279/5725	188802	5158200	249789	142	0
TBPC (800ps)	9-track	192178/0	192972	1925600	258989	-152	-12754
	12-track	0/181490	182290	2066200	325663	-79	-870
	mixed	177804/10640	189942	2272900	263435	73	0
Ratio	9-track			0.92	0.98		
	12-track			1.03	1.26		
	mixed			1.00	1.00		

VI. REGION UPDATE

In the region update stage, we identify regions that have cell overlapping, and enlarge their areas by a user-defined percentage β. To enlarge a region, we extend its height or width, depending on which dimension is smaller. In our current implementation, we set β as 10%.

VII. EXPERIMENTAL RESULTS

We have implemented our placement flow in a fully automated manner. The flow consists of our Tcl scripts, our point tools (for target cell clustering/re-clustering, pseudo net addition, region determination, and region update) written in C++, and a commercial placement tool. We conducted experiments in a TSMC 28nm technology node, and chose two different cell-height libraries that are 9-track and 12-track ones. We ran Synopsys Design Compiler [6] to synthesize three OpenCores [7] designs (AES, ECG, TBPC). Three different timing constraints were set for each design, and therefore a total of nine test cases were created. We used Cadence Innovus [3] as the placement engine in our flow and set the block utilization and aspect ration respectively to be 70% and 1 for generating the placement region. All the experiments were conducted on a machine with 2 Intel 1GHz cores and with 16 GB memory.

To demonstrate the effectiveness of our placement flow, we also synthesized the nine test cases into pure 9-track and 12-track gate-level netlists and placed them respectively using Synopsys Design Compiler and Cadence Innovus. Table I shows the statistics and results of each test case under different gate-level netlists. We observe that for the pure 9-track and 12-track test cases, seven and five placement results of theirs respectively fail to meet the given timing constraints. On the contrary, all the placement results produced by our flow are able to satisfy their timing constraints (i.e., their worst negative slack (WNS) values are all positive and their total negative slack (TNS) values are all zero). Our flow achieves 26% smaller block area and 3% less wirelength than the pure 12-track flow on average. Though the pure 9-track flow produces smaller block area and wirelength than our flow, majority of its placement results fail to meet the given timing constraints.

For each test case of the largest design (i.e., TBPC), the total runtime of our flow is about 3 to 6 hours. Nevertheless, the runtime spent by our point tools on clustering, region creation, and region update only accounts for 3.3% on average. Note that we cannot make a comparison with [2] because we do not have access to its code and test cases.

VIII. CONCLUSION

In this paper, we present a mixed-height standard cell placement flow. The efficacy of our flow is demonstrated by encouraging experimental results.

REFERENCES

[1] I. Markov, J. Hu, and M.-C. Kim, "Progress and challenges in vlsi placement research," in *Proc. ICCAD*, pp. 275–282, 2012.

[2] P. Gupta, A. B. Kahng, S. Nakagawa, S. Shah, and P. Sharma, "Mixed cell-height implementation for improved design quality in advanced nodes," in *Proc. ICCAD*, pp. 854–860, 2015.

[3] Cadence Innovus. http://www.cadence.com.

[4] S. v. Dongen, "Graph clustering by flow simulation," Ph.D. Thesis, University of Utrecht, 2000.

[5] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," in *IEEE Trans. Comput.*, vol. C-22, pp. 1025–1034, 1973.

[6] Synopsys Design Compiler. http://www.synopsys.com.

[7] OpenCores Design. http://opencores.org/.