Challenges and Solutions in Modern VLSI Placement

(Invited Paper)

Zhe-Wei Jiang[1], Hsin-Chen Chen[2], Tung-Chieh Chen[1], and Yao-Wen Chang[1,2]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan
{crazying, indark, donnie}@eda.ee.ntu.edu.tw; ywchang@cc.ee.ntu.edu.tw

***Abstract*— The VLSI placement problem is to place objects into a fixed die such that there are no overlaps among objects and some cost metric (e.g., wirelength, routability) is optimized. It is a major step in physical design that has been studied for decades. However, modern VLSI design challenges have reshaped the placement problem. A modern placer needs to handle large-scale designs with millions of objects, heterogeneous objects with very different sizes, and various complex placement constraints such as preplaced blocks and chip density. In this paper, we first introduce the major techniques employed in our placer for tackling the large-scale mixed-size designs and the aforementioned constraints, and then provide some future research directions for the modern placement problem.**

I. Introduction

The VLSI placement problem is to place objects into a fixed die such that there are no overlaps among objects and some cost metric (e.g., wirelength, routability) is optimized. It is a major step in physical design that has been studied for decades. Yet, it has attracted much attention recently because recent studies show that existing placers still cannot produce near optimal solutions. As a result, many new academic placers were invented in the recent years. Further, modern VLSI design challenges have reshaped the placement problem. As the feature size keeps shrinking, billions of transistors (or millions of standard cells) can be integrated in a single chip. Meanwhile, the intellectual property (IP) modules and pre-designed macro blocks (such as embedded memories, analog blocks, pre-designed datapaths, etc.) are often reused. As a result, advanced VLSI designs often contain a large number (hundreds) of macros of very different sizes from each other and the standard cells, and some of the macros may be preplaced in the chip. The dramatically increasing interconnect complexity further imposes routing difficulty. In addition to wirelength, therefore, modern placement shall also consider the density constraint.

To solve such the modern large-scale mixed-size placement problem, many academic placers were invented in recent years. Those placers can be classified into three major categories: (1) the analytical approach [3] [5] [11] [16] [22] [27], (2) the min-cut partitioning based approach [2] [4] [21], and (3) the hybrid approach [13] [25]. Among those approaches, the analytical placers have shown their superior efficiency and quality.

We shall first introduce the major techniques employed in the leading academic placer, NTUplace3 [5], which is based on the analytical technique and can handle modern large-scale mixed-size placement with wirelength, preplaced blocks, and density considerations. Like most modern placers, NTUplace3 consists of three major stages: global placement, legalization, and detailed placement. The global placement of NTUplace3 is based on the multilevel framework which applies a two-stage technique of bottom-up coarsening followed by top-down uncoarsening. The objective function is based on the log-sum-exp wirelength model [1] proposed by Naylor et al. [20]. To handle preplaced blocks, NTUplace3 applies a two-stage smoothing technique, Gaussian smoothing followed by level smoothing, to facilitate block spreading during global placement. The density is controlled mainly by cell spreading during global placement and cell sliding during detailed placement. We further use the conjugate gradient method with dynamic step-size control to speed up the global placement and apply macro shifting to find better macro positions.

During legalization, NTUplace3 removes the overlaps and places all standard cells into rows using a priority-based scheme based on block sizes and locations. A look-ahead legalization scheme is also incorporated into global placement to facilitate the legalization process. During detailed placement, NTUplace3 adopts cell matching and cell swapping to minimize the wirelength, and cell sliding to optimize the density.

Although the recent academic placers have made significant progress in the large-scale mixed-size placement problem, there are still many emerging challenges for this problem. As the number of macros increases dramatically, the single-stage methodology of integrated macro and standard-cell designs incurs significant difficulties in legality and complexity. Other design methodologies would be needed to tackle the increasing design complexity. In addition to wirelength, other cost metrics such as routability, timing, power, and thermal should also be addressed to handle the increasing integration complexity and operation frequency. We shall discuss these placement challenges and related research directions.

The remainder of this paper is organized as follows. Section II gives the analytical model used in NTUplace3. Section III explains the placement techniques employed in NTUplace3. Section IV reports the experimental results. Section V presents some future research directions. Finally, conclusions are given in Section VI.

II. Analytical Placement Model

Circuit placement can be formulated as a hypergraph $H = (V, E)$ placement problem. Let vertices $V = \{v_1, v_2, ..., v_n\}$ represent blocks and hyperedges $E = \{e_1, e_2, ..., e_m\}$ represent nets. Let x_i and y_i be the respective x and y coordinates of the center of the block v_i, and a_i be the area of the block v_i. The circuit may contain some preplaced blocks which have fixed x and y coordinates and are not movable. We intend to determine the optimal positions of movable blocks so that the total wirelength is minimized and there is no overlap among blocks.

x_i, y_i	center coordinate of block v_i
w_i, h_i	width and height of block v_i
w_b, h_b	width and height of bin b
M_b	maximum area of movable blocks in bin b
D_b	potential (area of movable blocks) in bin b
P_b	base potential (preplaced block area) in bin b
$t_{density}$	target placement density

Fig. 1. Notation used in this paper.

To evenly distribute the blocks, we divide the placement region into uniform non-overlapping bin grids. Then, the global placement problem can be formulated as a constrained minimization problem as follows:

$$\begin{array}{ll} \min & W(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} & D_b(\mathbf{x}, \mathbf{y}) \leq M_b, \quad \text{for each bin b,} \end{array} \tag{1}$$

where $W(\mathbf{x}, \mathbf{y})$ is the wirelength function, $D_b(\mathbf{x}, \mathbf{y})$ is the potential function that is the total area of movable blocks in bin b, and M_b is the maximum area of movable blocks in bin b. M_b can be computed by $M_b = t_{density}(w_b h_b - P_b)$, where $t_{density}$ is a user-specified target density value for each bin, w_b (h_b) is the width (height) of bin b, and P_b is the *base potential* that equals

[1]The log-sum-exp wirelength model is a patented technology [20] and use requires a license from Synopsys.

the preplaced block area in bin b. Note that M_b is a fixed value as long as all preplaced block positions are given and the bin size is determined. Figure 1 gives the notation used in this paper.

The wirelength $W(\mathbf{x},\mathbf{y})$ is defined as the total half-perimeter wirelength (HPWL) given by

$$W(\mathbf{x},\mathbf{y}) = \sum_{\text{net } e} \Big(\max_{v_i,v_j \in e} |x_i - x_j| + \max_{v_i,v_j \in e} |y_i - y_j| \Big). \quad (2)$$

Since $W(\mathbf{x},\mathbf{y})$ is non-convex, it is hard to minimize it directly. Thus, several smooth wirelength approximation functions are proposed in the literature. In NTUplace3, we apply the log-sum-exp wirelength model [20],

$$\gamma \sum_{e \in E} (\log \sum_{v_k \in e} \exp(x_k/\gamma) + \log \sum_{v_k \in e} \exp(-x_k/\gamma) + \log \sum_{v_k \in e} \exp(y_k/\gamma) + \log \sum_{v_k \in e} \exp(-y_k/\gamma)). \quad (3)$$

As $\gamma \approx 0$, log-sum-exp wirelength gives a good approximation to the HPWL [20].

The function $D_b(\mathbf{x},\mathbf{y})$ can be expressed as $D_b(\mathbf{x},\mathbf{y}) = \sum_{v \in V}^{n} P_x(b,v) P_y(b,v)$, where P_x and P_y are the overlap functions between bin b and block v along the x and y directions. Since density $D_b(\mathbf{x},\mathbf{y})$ is neither smooth nor differentiable, APlace [16] uses bell-shaped functions p_x and p_y for each block to smooth the density P_x and P_y, respectively. In [16], the bell-shaped potential function p_x is defined by

$$p_x(b,v) = \begin{cases} 1 - a d_x^2, & 0 \le d_x \le w_v/2 + w_b \\ b(d_x - 2w_b - 2w_g)^2, & w_v/2 + w_b \le d_x \le w_v/2 + 2w_b \\ 0, & w_v/2 + 2w_b \le d_x, \end{cases} \quad (4)$$

where

$$a = 4/((w_v + 2w_b)(w_v + 4w_b)) \\ b = 2/(w_b(w_v + 4w_b)). \quad (5)$$

Here, w_b is the bin width, w_v is the block width, and d_x is the x direction difference between the block v and the center of the bin b. The range of block's potential is $w_v + 2w_b$ in the x direction. The smooth y-potential function $p_y(b,v)$ can be defined similarly.

By doing so, the non-smooth function $D_b(\mathbf{x},\mathbf{y})$ can be replaced by the smooth one, $D'_b(\mathbf{x},\mathbf{y}) = \sum_{v \in V}^{n} c_v p_x(b,v) p_y(b,v)$, where c_v is a normalization factor so that the total potential of a block equals its area.

A quadratic penalty method is used to solve Equation (1), implying that we solve a sequence of unconstrained minimization problems of the form

$$\min \quad W(\mathbf{x},\mathbf{y}) + \lambda \sum_b (D'_b(\mathbf{x},\mathbf{y}) - M_b)^2 \quad (6)$$

with increasing λ's. The solution of the previous problem is used as the initial solution for the next one. We solve the unconstrained problem in Equation (6) by the conjugate gradient (CG) method. We observe that CG with line search in [16] is not efficient enough since the line search spends most running time on the minimization process. Therefore, we further use CG with a dynamic step size to minimize Equation (6). The dynamic step-size control, to be explained in the next Section, leads to significantly better efficiency.

III. Core Techniques of NTUplace3

Like many modern placers, NTUplace3 consists of three major steps: global placement, legalization, and detailed placement. Global placement evenly distributes the blocks and finds the better position for each block to minimize the target cost (e.g., wirelength). Then, legalization removes all overlaps among blocks and places standard cells into rows. Finally, detailed placement further refines the solution quality.

In the following sections, we describe the underlying techniques used in the global placement, legalization, and detailed placement of NTUplace3.

Algorithm: **Multilevel Global Placement**
Input:
hypergraph H_0: mixed-size circuit
n_{max}: the maximum block number in the coarsest level
Output:
(x^*, y^*): optimal block positions

01. $level$ = 0;
02. **while** $(BlockNumber(H_{level}) > n_{max})$
03. $level$++;
04. $H_{level} = FirstChoiceClustering(H_{level-1})$;
05. initialize block positions by $SolveQP(H_{level})$;
06. **for** $currentLevel$ = $level$ **to** 0
07. initialize bin grid size $n_{bin} \propto \sqrt{n_x}$;
08. initialize base potential for each bin;
09. initialize $\lambda_0 = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum |\partial D'_b(\mathbf{x},\mathbf{y})|}$; $m = 0$;
10. **do**
11. solve min $W(\mathbf{x},\mathbf{y}) + \lambda_m \sum (D'_b(\mathbf{x},\mathbf{y}) - M_b)^2$;
12. $m++$;
13. $\lambda_m = 2\lambda_{m-1}$;
14. **if** ($currentLevel$ == 0 & $overflow\_ratio < 10\%$)
15. call $LookAheadLegalization()$ and save the best result;
16. compute $overflow\_ratio$;
17. **until** (spreading enough or no further reduction in $overflow\_ratio$)
18. **if** ($currentLevel$ == 0)
19. restore the best look-ahead result;
20. **else**
21. call $MacroShifting()$;
22. decluster and update block positions.

Fig. 2. Our global placement algorithm.

A. Global Placement

As mentioned earlier, the global placement is based on the multilevel framework and the log-sum-exp wirelength model. A two-stage smoothing technique is used to handle preplaced blocks. We further use the conjugate gradient method with dynamic step-size control to speed up the global placement and apply macro shifting to find better macro positions. Now we detail those techniques.

1) Multilevel Framework: We use the multilevel framework for global placement to improve the scalability. Our algorithm is summarized in Figure 2. The coarsening stage (lines 1–4) iteratively clusters the blocks based on connectivity/size to reduce the problem size until a given threshold is reached. Then, we find an initial placement (line 5). In the uncoarsening stage (lines 6–22), it iteratively declusters the blocks and refines the block positions to reduce the objective function. The declustering process continues until all blocks are evenly distributed. In NTUplace3, the evenness of block distribution is measured by the *overflow ratio*, which is defined as follows:

$$overflow\_ratio = \frac{\sum_{\text{Bin } b} \max(D_b(\mathbf{x},\mathbf{y}) - M_b, 0)}{\sum \text{total movable area}}. \quad (7)$$

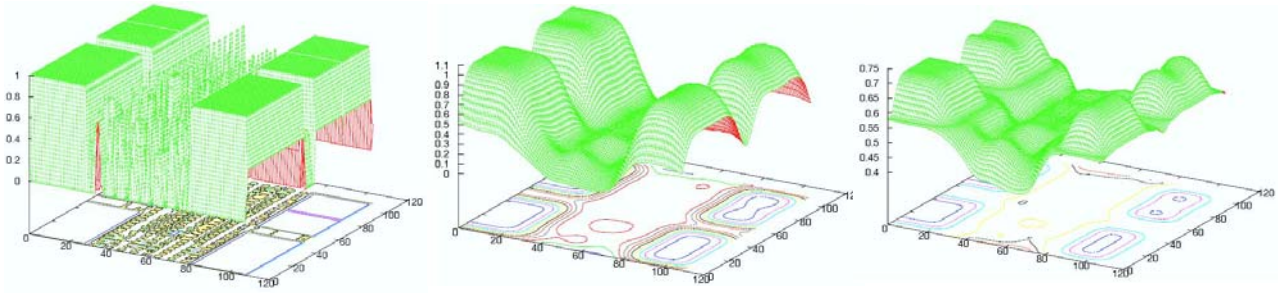
The global placement stage stops when the overflow ratio is less than a user-specified target value, which is 0 by default.

2) Base Potential Smoothing: Preplaced blocks pre-define the *base potential*, which significantly affects block spreading. Since the base potential P_b is not smooth, it incurs mountains that prevent movable blocks from passing through these regions. Therefore, we shall smooth the base potential to facilitate block spreading. We first use the Gaussian function to smooth the base potential change, remove the rugged regions in the base potential, and then smooth the base potential level so that blocks can spread to the whole placement region.

The base potential of each block can be calculated by the bell-shaped function. However, we observe that the potential generated by the bell-shaped function has "valleys" among the adjacent regions of blocks, and these regions do not have any free space but their potentials are so low that a large number of blocks may spread to these regions. To avoid this problem, we use the Gaussian function to smooth the base potential. The two-dimensional Gaussian is given by

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (8)$$

where σ is the standard deviation of the distribution. Applying convolution to the Gaussian function G with the base potential P, $P'(x,y) = G(x,y) * P(x,y)$, we can obtain a smoother base



(a) (b) (c)

Fig. 3. (a) The density profile of newblue2. (b) The base potential after Gaussian smoothing results in a better smoothing potential. (c) The base potential after level smoothing.

potential P'. Gaussian smoothing works as a low-pass filter, which can smooth the local density change.

After the Gaussian smoothing, we apply another landscape smoothing function [8] [14] to reduce the potential levels. The smoothing function $P''(x, y)$ is defined as follows:

$$P''(x,y) = \begin{cases} \overline{P'} + (P'(x,y) - \overline{P'})^{\delta} & \text{if } P'(x,y) \geq \overline{P'} \\ \overline{P'} - (\overline{P'} - P'(x,y))^{\delta} & \text{if } P'(x,y) \leq \overline{P'}, \end{cases} \quad (9)$$

where $\delta \geq 1$. Level smoothing reduces "mountain" (high potential regions) heights so that blocks can spread to the whole placement area smoothly. Figure 3 shows the smoothing process of the circuit newblue2.

3) Conjugate Gradient Search with Dynamic Step Size: We use the conjugate gradient (CG) method with dynamic step size instead of line search to minimize Equation (6). After computing the conjugate gradient direction d_k, the step size α_k is computed by $\alpha_k = s/||d_k||_2$, where s is a user-specified scaling factor. By doing so, we can limit the step size of block spreading since the total quadratic Euclidean movement is fixed,

$$\sum_{v_i \in V} (\Delta x_i^2 + \Delta y_i^2) = ||\alpha_k d_k||_2^2 = s^2, \quad (10)$$

where Δx_i and Δy_i are the respective amounts of the movement along the x and y directions for the block v_i in each iteration.

To show the effectiveness of the dynamic step-size control, we performed experiments on adaptec1 with different step sizes. In Figure 4, the CPU times and HPWLs are plotted as functions of the step sizes. As shown in Figure 4, the CPU time decreases as the step size s becomes larger. In contrast, the HPWL decreases as the step size s gets smaller. The results show that the step size significantly affects the running time and the solution quality.

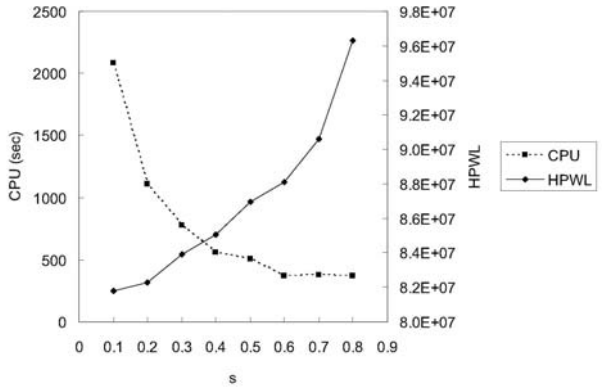


Fig. 4. The CPU times and HPWLs resulting from different step sizes based on the circuit adaptec1.

4) Macro Shifting: In the global placement stage, it is important to preserve legal macro positions since macros are much bigger than standard cells and illegal macro positions typically make legalization much more difficult. To avoid this, we apply macro shifting at each declustering level of the global placement stage. Macro shifting moves macros to the closest legal positions. Integrated within the multilevel framework, only macros with sizes larger than the average cluster size of the current level are processed. Then, the legal macro positions provide a better initial solution for the next declustering level, and those macros are still allowed to spread at subsequent declustering levels.

B. Legalization

After global placement, legalization removes all overlaps and places standard cells into rows. We extend the standard-cell legalization method in [10] to solve the mixed-size legalization problem. The legalization order of macros and cells are determined by their x coordinates and sizes (widths and heights). Larger blocks get the priority for legalization. Therefore, we legalize macros earlier than standard cells. After the legalization order is determined, macros are placed to their nearest available positions and cells are packed into rows with the smallest wirelength. Despite its simplicity, this macro/cell legalization strategy works well on all benchmarks.

Recall that we performed block spreading during global placement. It is important to determine when to terminate the block spreading. If blocks do not spread enough, the wirelength may significantly be increased after legalization since blocks are over congested. If blocks spread too much, the wirelength before legalization may not be good even the legalization step only increases wirelength a little. This situation becomes even worse when the density is also considered, since the placement objective is more complicated.

To improve the legalization quality, we use a look-ahead legalization technique during global placement to make the subsequent legalization process easier. At the finest level of the multilevel placement, we apply legalization right after placement objective optimization in each iteration and record the best result with the minimum cost (wirelength and density penalty). Although the look-ahead legalization may take longer running time due to more iterations of legalization, we can ensure that blocks do not over spread and thus obtain a better legal placement. As a result, the look-ahead legalization significantly alleviates the difficulty in removing the macro and standard-cell overlaps during the later legalization stage, and eventually leads to a more robust placement result.

C. Detailed Placement

The detailed placement stage consists of two stages: the wirelength minimization stage and the density optimization stage. In the wirelength minimization stage, we apply *cell matching* and *cell swapping* to reduce the total wirelength. In the density optimization stage, we apply the *cell sliding* technique to reduce the density overflow in congested regions. In the following, we explain the cell-matching, cell-swapping, and cell-sliding algorithms.

1) Cell Matching: We extend the window-based detailed placement (WDP) algorithm [13] and name our approach *cell matching* here. The WDP algorithm finds a group of exchangeable cells inside a given window, and formulates a bipartite matching problem by matching the cells to the empty slots in the window. The cost is given by the HPWL difference of a cell in each empty slot. The bipartite matching problem can be solved optimally in polynomial time, but the optimal assignment cannot guarantee the optimal HPWL result because the HPWL cost of a cell to each empty slot depends on the positions of the other connected cells. Our cell matching algorithm remedies this drawback by selecting *independent cells* at one time to perform bipartite matching. Here by independent cells, we mean that there is no common net between any pairs of the selected cells.

2) Cell Swapping: The cell swapping technique selects k adjacent cells each time to find the best ordering by enumerating all possible orderings using the branch-and-bound method. Here, k is a user-specified parameter. In our implementation, we set $k = 3$ for a good tradeoff between the running time and solution quality. This process repeats until all standard cells are processed.

3) Cell Sliding: The objective of cell sliding is to reduce the density overflow in the congested area. We divide the placement region into uniform non-overlapping bins, and then iteratively reduce the densities of overflowed bins by sliding the cells horizontally from denser bins to sparser bins, with the cell order being preserved. Each iteration consists of two stages: left sliding and right sliding. In each stage, we calculate the density of each bin and then compute the area flow $f_{bb'}$ between bin b and its left or right neighboring bin b'. Here, $f_{bb'}$ denotes the desired amount of cell area to move from bin b to b'. Recall that we define D_b as the total movable cell area in bin b and M_b as the maximum allowable block area in bin b. If bin b has no area overflow or the area overflow ratio of b is smaller than b', that is $D_b \leq M_b$ or $D_b/M_b \leq D_{b'}/M_{b'}$, we set $f_{bb'} = 0$. Otherwise we calculate $f_{bb'}$ according to the capacity of b'. If bin b' has enough free space, we move the overflow area of bin b to b'. Otherwise, we evenly distribute the overflow area between b and b'. Therefore, $f_{bb'}$ is defined by

$$f_{bb'} = \begin{cases} D_b - M_b, & if\ (M_{b'} - D_{b'}) \geq (D_b - M_b) \\ \frac{D_b M_{b'} - D_{b'} M_b}{M_b + M_{b'}}, & otherwise, \end{cases} \tag{11}$$

where the second condition of Equation (11) is derived from

$$D_b - \left(M_b + \frac{(D_b - M_b + D_{b'} - M_{b'})M_b}{M_b + M_{b'}} \right) = \frac{D_b M_{b'} - D_{b'} M_b}{M_b + M_{b'}}. \tag{12}$$

After the area flow $f_{bb'}$ is computed, we sequentially slide the cells across the boundary between b and b' until the amount of sliding area reaches $f_{bb'}$ or there is no more area for cell sliding. Then we update D_b and $D_{b'}$. In the right sliding stage, we start from the left-most bin of the placement region, and b' is right to b. In the left sliding stage, we start from the right-most bin, and b' is left to b, accordingly. We iterative slide the cells from the area overflow regions to sparser regions until no significant improvement can be obtained.

IV. Experimental Results

The experiment was performed on a Linux PC with an AMD Opteron 2.6GHz CPU. Table I gives the normalized average HPWL, DHPWL, CPU time, and score of NTUplace3 and other state-of-the-art academic placers based on the ISPD'06 placement contest benchmark suite [1] [19]. The density HPWL (DHPWL) and the score function are defined as follows:

$$DHPWL = HPWL \times (1 + density\_penalty),$$
$$Score = HPWL \times (1 + density\_penalty + cpu\_factor).$$

Among all placers, we obtained both the best average HPWL and the best average DHPWL. Further, according to the scoring function in the 2006 ISPD Placement Contest, placers with 2X (4X) CPU time incur about 4% (8%) penalty. Therefore, our overall result considering (1) HPWL, (2) density penalty, and (3) the CPU factor, is the best among all participating placers, and is about 4%, 5%, and 6% better than the three leading placers, Kraftwerk [22], mPL6 [3], and NTUplace2 respectively. See Figure 5 for the placement result of the circuit newblue7.

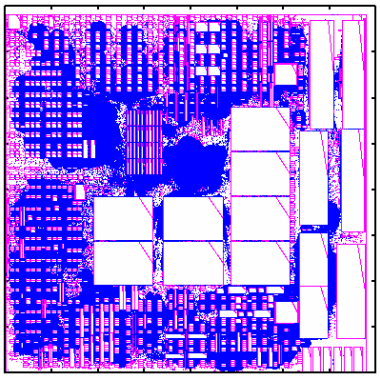


Fig. 5. A resulting placement of the circuit newblue7.

TABLE I

HPWL, DHPWL, CPU Time, and Score comparison based on the ISPD'06 benchmarks

Placer	HPWL	DHPWL	CPU time	Score
NTUplace3 [5]	**1.00**	**1.00**	1.00	**0.99**
Kraftwerk [22]	1.12	1.08	0.59	1.03
mPL6 [3]	1.06	1.01	2.08	1.04
NTUplace2 [13]	1.04	1.02	1.92	1.05
mFAR [11]	1.14	1.10	1.38	1.11
APlace3 [17]	1.13	1.10	3.64	1.16
Dragon [25]	1.36	1.29	**0.51**	1.23
FastPlace [27]	1.21	1.38	0.58	1.33
DPlace	1.37	1.41	0.75	1.36
Capo [21]	1.41	1.34	2.20	1.39

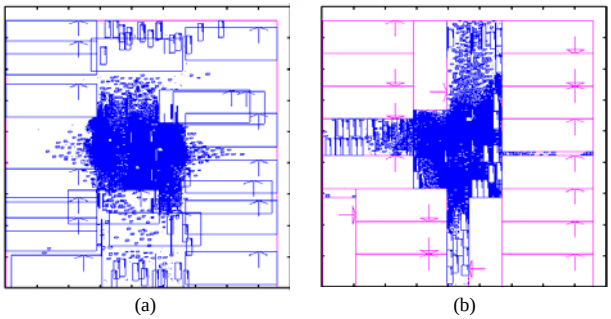
V. Future Challenges

Although recent academic placers (e.g., NTUplace3) have made significant progress in the large-scale mixed-size placement problem, there are still many emerging challenges arising from advanced VLSI process technologies. In this section, we present some potential research directions for modern VLSI placement.

A. Macro Placement

We can classify the methods for handling the large-scale mixed-size designs into three major categories: (1) simultaneous macro and standard-cell placement: Most existing mixed-sized placers (e.g., APlace, Kraftwerk, mPL, NTUplace3, etc.) employ this single-stage methodology for macro and standard-cell placement. As the number of macros increases dramatically due to the pervasive use of IP modules, however, the methodology incurs significant difficulties in legality and complexity. Consequently, a robust legalizer is desirable for this method. (2) constructive macro placement: Most partitioning based placers (e.g., Capo, PATOMA [7]) keep macro overlap-free during the placement process by recursively partitioning the chip/macros into subregions. An intrinsic limitation of this hierarchical approach lies in the lack of the global interaction among different subregions/macros, and thus the solution quality is also limited, especially for the placement instances with low utilization rates. (3) two-stage macro placement: The two-staged approach consists of macro placement followed by standard-cell placement. This approach is more robust in finding legal placement, and is thus widely used in the industry. For this approach, it is often needed to obtain a prototype of macro placement with the consideration of standard-cell positions to guide the legalization of the macros. It is thus desirable to develop an effective macro placement algorithm that can consider the interactions among macros as well as between macros and standard cells. See Figure 6(a) for an illegal placement for the circuit newblue3 and Figure 6(b) for a legal placement with the special consideration of macros.

Further, there are many other issues that need to be considered for real-world macro placement. For example, placement blockages and pre-placed macros, performance, hierarchy, and region constraints, macro orientations, and (power) pin positions,



(a) (b)

Fig. 6. (a) An illegal placement of the circuit newblue3. (b) A legal placement generated by NTUplace3 with the special consideration of macros.

and so on. There are substantial research opportunities along this direction.

B. Routability-Driven Placement

Most existing placement algorithms focus on total wirelength minimization to obtain better circuit performance and smaller layout area. Despite the pervasive use of the half-perimeter wirelength objective, there is a significant mismatch between wirelength and congestion objectives in placement. Although most routing algorithms can handle congestion, often routing congestion violations cannot be totally removed if the given placement does not consider routability. Therefore, it is of particular significance to consider routability during placement, especially for modern VLSI designs with very large-scale interconnections.

Previous works [16] [18] [28] allocate white spaces into congested regions for better routability. However, persevering white spaces does not solve the congestion problem effectively. Another issue is that, due to the lack of interactions between placement and routing, routers may not honor the resource allocation obtained from placers. Thus, a potential research direction is to develop a fast and accurate routing demand estimation and incorporate it into the placement stage, or even a simultaneous placement and routing algorithm for modern VLSI designs if the time complexity is tractable.

C. Timing-Driven Placement

In high-performance circuits, a large portion of timing optimization is performed in the placement stage. Traditional placement algorithms usually achieve the timing goal via wirelength minimization. Nevertheless, there is a significant gap between wirelength and actual delay, so many methods have been proposed recently to tackle this challenge. Those proposed timing-driven placement methods can be classified into two major categories: (1) path-based and (2) net-based methods. The path-based methods [9] [12] [23] [24] try to control critical path delays directly, but they might incur prohibitively high time complexity for modern large-scale circuits due to their exponentially growing numbers of paths. The net-based method [15], in contrast, transfers the timing constraint of each path into net weights. However, since the net-based method ignores the global views of a full path, it is less accurate for the timing control. Due to the limitations in existing timing-driven placement algorithms, it is desirable to further explore the timing optimization techniques with lower complexity and higher controllability for placement.

D. Power-Aware Placement

Power consumption has long become the first-order cost metric not only for hand-held devices for longer battery life, but also for high-performance applications for lower heat dissipation (and thus cooling cost). Previous works, like Cheon *et al.* [6], proposed to reduce the power consumption during the placement stage. Moreover, to further reduce the power consumption under the performance constraint, multiple supply voltages (MSV's) [26] are widely used for low-power designs, and bring up many new research opportunities in various design stages. Integration of voltage assignment and placement is essential to further reduce the power consumption for MSV designs.

E. Thermal Placement

As the process technology advances, the feature size keeps shrinking and thus the integration density keeps increasing while the clock frequency keeps rising. As a result, the increased power density significantly raises the chip temperature. However, reducing the power consumption alone is not sufficient to reduce the chip temperature, since the power density is also a dominant factor. Therefore, it is desirable to develop new techniques that can evenly spread hot blocks/cells over the whole placement region to lower the chip temperature and increase the chip reliability.

VI. Conclusions

Modern VLSI design challenges have reshaped the placement problem. In this paper, we have presented example techniques to tackle the challenges arising from large-scale mixed-size circuit designs with the wirelength optimization. Although significant progress has been made in placement research, modern circuit designs have induced many more challenges and opportunities for future research on macro placement and routability-, timing-, power-, and/or thermal-driven optimization of the placement problem.

VII. Acknowledgments

This work was partially supported by National Science Council of Taiwan under Grant No's NSC 95-2221-E-002-372, NSC 95-2221-E-002-374, and NSC 95-2752-E-002-008-PAE.

References

[1] *ISPD 2006 Program*. http://www.ispd.cc/program.html.

[2] A. R. Agnihotri, S. Ono, and P. H. Madden. Recursive bisection placement: Feng Shui 5.0 implementation details. In *Proc. of ISPD*, pages 230–232, 2005.

[3] T. Chan, J. Cong, J. Shinnerl, K. Sze, and M. Xie. mPL6: Enhanced multilevel mixed-size placement. In *Proc. of ISPD*, pages 212–214, 2006.

[4] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang, and Y.-W. Chang. NTUplace: a ratio partitioning based placement algorithm for large-scale mixed-size designs. In *Proc. of ISPD*, pages 236–238, 2005.

[5] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. A high-quality mixed-size analytical placer considering preplaced blocks and density constraints. In *Proc. of ICCAD*, 2006.

[6] Y. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang. Power-aware placement. In *Proc. of DAC*, pages 795–800, 2005.

[7] J. Cong, M. Romesis, and J. R. Shinnerl. Fast floorplanning by look-ahead enabled recursive bipartitioning. In *Proc. of ASPDAC*, 2005.

[8] J. Gu and X. Huang. Efficient local search with search space smoothing: A case study of the traveling salesman problem (TSP). *Trans. on SMC*, 24(5):728–735, 1994.

[9] T. Hamada, C. K. Cheng, and P. M. Chau. Prime: A placement tool using a piece wise linear resistive network approach. In *Proc. of DAC*, pages 531–536, 1993.

[10] D. Hill. US patent 6,370,673: Method and system for high speed detailed placement of cells within an intergrated circuit design. 2002.

[11] B. Hu, Y. Zeng, and M. Marek-Sadowska. mFAR: fixed-points-addition-based VLSI placement algorithm. In *Proc. of ICCAD*, pages 239–241, 2006.

[12] M. Jackson and E. S. Kuh. Performance-driven placement of cell based ic's. In *Proc. of DAC*, pages 370–375, 1989.

[13] Z.-W. Jiang, T.-C. Chen, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. NTUplace2: A hybrid placer using partitioning and analytical techniques. In *Proc. of ISPD*, pages 215–217, 2006.

[14] A. B. Kahng, S. Reda, and Q. Wang. APlace: A general analytic placement framework. In *Proc. of ISPD*, pages 233–235, 2005.

[15] A. B. Kahng and Q. Wang. An analytic placer for mixed-size placement and timing-driven placement. In *Proc. of ICCAD*, pages 565–572, 2004.

[16] A. B. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. *IEEE Trans. on CAD*, 24(5), May 2005.

[17] A. B. Kahng and Q. Wang. A faster implementation of APlace. In *Proc. of ISPD*, pages 218–220, 2006.

[18] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden. Routability-driven placement and white space allocation. In *Proc. of ICCAD*, pages 394–401, 2004.

[19] G.-J. Nam, C. J. Aplert, and P. G. Villarrubia. The ISPD 2006 placement contest and benchmark suite. In *Slides presented at ISPD'06*, 2006.

[20] W. C. Naylor, R. Donelly, and L. Sha. US patent 6,301,693: Non-linear optimization system and method for wire length and dealy optimization for an automatic electric circuit placer. 2001.

[21] J. Roy, D. Papa, A. Ng, and I. Markov. Satisfying whitespace requirements in top-down placement. In *Proc. of ISPD*, pages 206–208, 2006.

[22] P. Spindler and F. M. Johannes. Fast and robust quadratic placement combined with an exact linear net model. In *Proc. of ICCAD*, 2006.

[23] A. Srinivasan, K. Chaudhary, and E. S. Kuh. Ritual: Performance driven placement algorithm for small cell ics. In *Proc. of ICCAD*, pages 48–51, 1991.

[24] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *Proc. of DAC*, pages 211–215, 1995.

[25] T. Taghavi, X. Yang, B.-K. Choi, M. Wang, and M. Sarrafzadeh. Dragon2006: Blockage-aware congestion-controlling mixed-size placer. In *Proc. of ISPD*, pages 209–211, 2006.

[26] K. Usami and M. Horowitz. Clustered voltage scaling technique for low-power design. In *Proc. of ISQED*, pages 3–8, 1995.

[27] N. Viswanathan, M. Pan, and C. Chu. FastPlace 2.0: An efficient analytical placer for mixed-mode designs. In *Proc. of ASPDAC*, pages 195–200, 2006.

[28] X. Yang, B.-K. Choi, and M. Sarrafzadeh. Routability-driven white space allocation for fixed-die standard-cell placement. In *Proc. of ISPD*, pages 42–47, 2002.