NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs With Preplaced Blocks and Density Constraints

Tung-Chieh Chen, *Student Member, IEEE*, Zhe-Wei Jiang, *Student Member, IEEE*, Tien-Chang Hsu, Hsin-Chen Chen, *Student Member, IEEE*, and Yao-Wen Chang, *Member, IEEE*

***Abstract*—In addition to wirelength, modern placers need to consider various constraints such as preplaced blocks and density. We propose a high-quality analytical placement algorithm considering wirelength, preplaced blocks, and density based on the log-sum-exp wirelength model proposed by Naylor *et al.* and the multilevel framework. To handle preplaced blocks, we use a two-stage smoothing technique, i.e., Gaussian smoothing followed by level smoothing, to facilitate block spreading during global placement (GP). The density is controlled by white-space reallocation using partitioning and cut-line shifting during GP and cell sliding during detailed placement. We further use the conjugate gradient method with dynamic step-size control to speed up the GP and macro shifting to find better macro positions. Experimental results show that our placer obtains very high-quality results.**

***Index Terms*—Legalization (LG), physical design, placement.**

I. Introduction

As process technology advances, the feature size is getting smaller and smaller. As a result, billions of transistors can be integrated in a single chip. Meanwhile, the intellectual property modules and predesigned macro blocks (such as embedded memories, analog blocks, predesigned datapaths, etc.) are often reused. As a result, modern advanced IC designs often contain millions of standard cells and hundreds of macros with different sizes. Hence, modern placers need to handle the instances with large-scale mixed-size macros and standard cells.

Manuscript received September 17, 2006; revised August 18, 2007. This work was supported in part by MediaTek Inc., National Science Council of Taiwan, R.O.C., under Grant NSC 94-2215-E-002-030 and Grant NSC 94-2752-E-002-008-PAE, and in part by RealTek Semiconductor Corporation. This paper was recommended by Associate Editor C. J. Alpert.

T.-C. Chen is with the Graduate Institute of Electronic Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. He is also with SpringSoft, Inc., Hsinchu 300, Taiwan, R.O.C. (e-mail: donnie@eda.ee.ntu.edu.tw).

Z.-W. Jiang is with the Graduate Institute of Electronic Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: crazying@eda.ee.ntu.edu.tw).

T.-C. Hsu was with the Graduate Institute of Electronic Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. He is now with Synopsys Taiwan Ltd., Taipei 110, Taiwan, R.O.C. (e-mail: tchsu@eda.ee.ntu.edu.tw).

H.-C. Chen was with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. He is currently serving in the military in Taiwan, R.O.C. (e-mail: indark@eda.ee.ntu.edu.tw).

Y.-W. Chang is with the Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C., and also with Waseda University, Tokyo 169-8050, Japan (e-mail: ywchang@cc.ee.ntu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2008.923063

In addition, high-performance IC designs usually require significant white space for further performance optimization, such as buffer insertion and gate sizing. Therefore, density control and white-space allocation (WSA) have become very important. A wirelength-driven placer without considering placement density tends to pack blocks together to minimize wirelength. However, an overcongested region may not have enough white space for buffer insertion and thus degrade the chip performance. Although some congestion-aware placement algorithms were proposed [3], [4], these algorithms intend to minimize the routing congestion, which is different from the density control since the density can still be high for some regions even if no routing overflows occur in those regions.

Further, modern chip designs often consist of many preplaced blocks, such as analog blocks, memory blocks, and/or I/O buffers, which are fixed in the chip and cannot overlap with other blocks. These preplaced blocks impose more constraints on the placement problem. A placement algorithm without considering preplaced blocks may generate illegal placement or inferior solutions.

Most of the recently proposed placement algorithms can handle the mixed-size constraints [5]–[11]. However, very few modern mixed-size placement algorithms can satisfactorily handle preplaced blocks and the chip density. In this paper, we present a high-quality mixed-size analytical placement algorithm considering preplaced blocks and density constraints. Our placer is based on a three-stage technique: 1) global placement (GP); 2) legalization (LG); and 3) detailed placement (DP). It has the following distinguished features.

1) Based on the log-sum-exp wirelength model[1] proposed by Naylor *et al.* [2] and the multilevel framework, our placer consistently generates high-quality mixed-size placement results.
2) To solve the unconstrained minimization placement problem, we use the conjugate gradient (CG) method with dynamic step sizes. Experimental results show that the method leads to significant run-time speedups.
3) Our placer handles preplaced blocks by a two-stage smoothing technique. The preplaced block potential is first smoothed by a Gaussian function to remove the rugged potential regions, and then the potential levels are

[1]The log-sum-exp wirelength model is a patented technology [2], and use requires a license from Synopsys.

TABLE I
COMPARISONS BETWEEN OUR PLACER AND APLACE AND MPL; ALL THE PLACERS ARE BASED ON THE ANALYTICAL TECHNIQUE AND THE LOG-SUM-EXP WIRELENGTH MODEL. UNKNOWN: NOT MENTIONED IN THE CORRESPONDING WORK

	APlace 2.0/3.0	mPL5/mPL6	Ours
Global Placement Framework	V-cycle multilevel framework	W-cycle multilevel framework (mPL5) V-cycle multilevel framework (mPL6)	V-cycle multilevel framework
Clustering	Best-choice clustering	First-choice clustering (mPL5) Best-choice clustering (mPL6)	First-choice clustering
Wirelength Model	Log-sum-exp	Log-sum-exp	Log-sum-exp
Spreading Force	Bell-shaped potential	Poisson smoothed potential	Bell-shaped potential
Nonlinear Objective Solver	Conjugate gradient method w/ golden section line search	Explicit Euler method	Conjugate gradient method w/ dynamic step-size control
Preplaced Block Handling	Level smoothing	Poisson equation smoothing	Gaussion and level smoothing
Density Handling	Unknown	Network-flow-based cell redistribution	White-space allocation, Cell sliding
Macro Block Handling	Unknown	Linear programming based macro legalization	Macro shifting
Look-Ahead Legalization	No	No	Yes

smoothed so that movable blocks can effectively spread to the whole placement region.

4) Density constraints are considered during both GP and DP. We reallocate the white space using partitioning and cut-line shifting to remove density overflows between different levels of GP. In DP, a cell-sliding technique is applied to reduce the density overflow.
5) A macro shifting technique is used between levels of GP to find better macro positions that are easier for LG.
6) A look-ahead LG scheme during GP is used to obtain a better legal placement result. The legalizer is called several times near the end of GP. This technique can reduce the gap between GP and LG.

Table I summarizes the comparisons between our placer and two state-of-the-art analytical placers, i.e., APlace 2.0/3.0 [12], [13] and mPL5/6 [6], [14], which are also based on the log-sum-exp wirelength model. In the table, "Unknown" denotes that the corresponding method is not available in the literature.

The remainder of this paper is organized as follows. Section II gives the analytical model used in our placer. Our core placement techniques are explained in Section III. Section IV reports the experimental results. Finally, the conclusions are given in Section V.

II. ANALYTICAL PLACEMENT MODEL

The circuit placement problem can be formulated as a hypergraph $H = (V, E)$ placement problem. Let the vertices $V = \{v_1, v_2, \ldots, v_n\}$ represent blocks and the hyperedges $E = \{e_1, e_2, \ldots, e_m\}$ represent nets. Let x_i and y_i be the x and y coordinates of the center of block v_i, and let a_i be the area of the block v_i. The circuit may contain some *preplaced blocks* that have fixed x and y coordinates and cannot be moved. We intend to determine the optimal positions of movable blocks so that the total wirelength is minimized and there is no overlap among blocks. The placement problem is usually solved in three stages: 1) GP; 2) LG; and 3) DP. GP evenly distributes the blocks and finds the best position for each block to minimize the target cost (e.g., wirelength). Then, LG removes all overlaps. Finally, DP refines the solution.

Fig. 1 shows the notation used in this paper.

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
P_b	base potential (area of preplaced blocks) in bin b
D_b	potential (area of movable blocks) in bin b
M_b	the maximum potential 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 nonoverlapping bin grids. Then, the GP problem can be formulated as a constrained minimization problem as follows:

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

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 allowable area of movable blocks in bin b. M_b can be computed by

$$M_b = t_{\text{density}}(w_b h_b - P_b) \tag{2}$$

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 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.

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

$$W(\mathbf{x}, \mathbf{y}) = \sum_{\text{net } e} \left(\max_{v_i, v_j \in e} |x_i - x_j| + \max_{v_i, v_j \in e} |y_i - y_j| \right). \tag{3}$$

Since $W(\mathbf{x}, \mathbf{y})$ is not smooth and nonconvex, it is hard to directly minimize it. Thus, several smooth wirelength approximation functions are proposed, such as quadratic wirelength

[15], [16], L_p-norm wirelength [13], [14], and log-sum-exp wirelength [2], [5], [6]. The log-sum-exp wirelength model

$$\gamma \sum_{e\in E} \left((\log \sum_{v_k\in e} \exp(x_k/\gamma) + \log \sum_{v_k\in e} \exp(-x_k/\gamma) + \log \sum_{v_k\in e} \exp(y_k/\gamma) + \log \sum_{v_k\in e} \exp(-y_k/\gamma) \right) \quad (4)$$

proposed in [2] achieves the best result among these three models [14]. When γ is small, the log-sum-exp wirelength is close to the HPWL [2]. However, due to the computer precision, we can only choose a reasonably small γ, for example, 1% long of the chip width, so that it will not cause any arithmetic overflow.

Since the density $D_b(\mathbf{x},\mathbf{y})$ is neither smooth nor differentiable, mPL [14] uses inverse Laplace transformation to smooth the density, whereas APlace [5] uses a bell-shaped function for each block to smooth the density. We express the function $D_b(\mathbf{x},\mathbf{y})$ as

$$D_b(\mathbf{x},\mathbf{y}) = \sum_{v\in V} P_x(b,v)P_y(b,v) \quad (5)$$

where P_x and P_y are the overlap functions of bin b and block v along the x- and y-directions. We adopt the bell-shaped potential function [5] p_x to smooth P_x. p_x is defined by

$$p_x(b,v) = \begin{cases} 1 - ad_x^2, & 0 \le d_x \le \frac{w_v}{2} + w_b \\ b\left(d_x - \frac{w_v}{2} - 2w_b\right)^2, & \frac{w_v}{2} + w_b \le d_x \le \frac{w_v}{2} + 2w_b \\ 0, & \frac{w_v}{2} + 2w_b \le d_x \end{cases} \quad (6)$$

where

$$a = \frac{4}{(w_v + 2w_b)(w_v + 4w_b)}$$
$$b = \frac{2}{w_b(w_v + 4w_b)} \quad (7)$$

w_b is the bin width, w_v is the block width, and d_x is the center-to-center distance of the block v and the bin b in the x-direction. Fig. 2(a) and (b) shows the original and smoothed overlap functions, respectively. The range of the block's potential is $w_v + 4w_b$ in the x-direction. The smooth y-potential function $p_y(b,v)$ can be defined in a similar way, and the range of the block's potential is $h_v + 4h_b$ in the y-direction. By doing so, the nonsmooth function $D_b(\mathbf{x},\mathbf{y})$ can be replaced by a smooth one

$$\hat{D}_b(\mathbf{x},\mathbf{y}) = \sum_{v\in V}^{n} c_v p_x(b,v)p_y(b,v) \quad (8)$$

where c_v is a normalization factor so that the total potential of a block equals its area.

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

$$\min \quad W(\mathbf{x},\mathbf{y}) + \lambda \sum_b \left(\hat{D}_b(\mathbf{x},\mathbf{y}) - M_b\right)^2 \quad (9)$$

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 (9) by the CG method. Further, we observe that CG with line search in [5] is not efficient since the line-search method takes most portion of its runtime for the minimization process. Therefore, we use CG with a dynamic step size to minimize (9). Numerical results show that our approach is much faster than that used in [5].

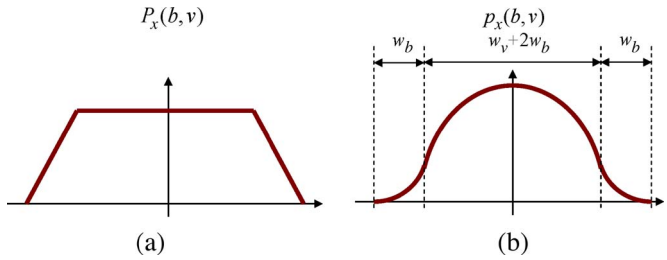

Fig. 2. (a) Overlap function $P_x(b,v)$. (b) Smoothed overlap function $p_x(b,v)$.

III. Proposed Algorithm

As mentioned earlier, our placement algorithm consists of three stages: 1) GP; 2) LG; and 3) DP. We detail each stage in the following sections.

A. GP

Our placement algorithm is based on the aforementioned analytical technique and the multilevel framework. The multilevel framework adopts a two-stage flow of clustering followed by declustering. At each level of declustering, GP is performed to find the best positions for macros and standard cells. For the analytical search, the CG search with dynamic step-size control is adopted to speed up the search for a desirable solution. To handle preplaced blocks, we resort to a two-stage smoothing technique of Gaussian smoothing followed by level smoothing to smooth the search space and thus facilitate cell spreading. To control the chip density, we apply white space distribution to allocate more white space to areas with density overflows. To facilitate macro and cell LG, we further apply macro shifting and look-ahead LG (described in Section III-B2) at the GP stage. We detail the aforementioned techniques in the following sections.

1) Multilevel Framework: We use the multilevel framework for GP to improve the scalability. Our algorithm is summarized in Fig. 3. Lines 1–4 are the coarsening stage. The initial placement is generated in line 5. Lines 6–23 are uncoarsening stages. The details of each step are explained as follows.

During the coarsening stage, we cluster blocks to reduce the number of movable blocks. The hierarchy of clusters is built by the first-choice (FC) clustering algorithm [14]. To apply the

Algorithm: **Multilevel Global Placement**
Input:
hypergraph H_0: a mixed-size circuit
n_{max}: the maximum block number in the coarsest level
Output:
(x^*, y^*): optimal block positions without considering block overlaps

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;
08. initialize base potential for each bin;
09. initialize $\lambda_0 = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum |\partial \hat{D}_b(\mathbf{x},\mathbf{y})|}$; $m = 0$;
10. **do**
11. solve min $W(\mathbf{x},\mathbf{y}) + \lambda_m \sum (\hat{D}_b - M_b)^2$;
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. call $WhiteSpaceAllocation()$;
23. decluster and update block positions.

Fig. 3. Our multilevel GP algorithm.

FC clustering algorithm, we examine each block in the circuit one-by-one, find the block with the highest connectivity, and cluster these two blocks. We control the area of a clustered block so that it will not be 1.5 times larger than the average area of clustered blocks. The clustering process continues until the number of blocks is reduced by five times, and then we obtain a level of clustered circuit. The FC clustering algorithm is applied several times until the block number in the resulting clustered circuit is less than a user-specified number $n_{\max}$, for example, 6000 by default.

After clustering, the initial placement for the coarsest level is generated by minimizing the quadratic wirelength using the CG method, the same method as in quadratic placement.

Then, we solve the placement problem from the coarsest level to the finest level. The placement for the current level provides the initial placement for the next level. The horizontal/vertical grid numbers are set to the square root of the number of clusters in the current level, i.e., $grid\_num\_v = grid\_num\_h = \sqrt{BlockNumber(H_{level})}$. Then, the base potential P_b for each bin is computed, and the maximum area of movable blocks M_b is updated accordingly. In addition, the value of λ is initialized according to the strength of wirelength and density gradients as

$$\lambda = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum \left|\partial \hat{D}_b(\mathbf{x},\mathbf{y})\right|} \tag{10}$$

and the value of λ is increased by two times for each iteration. A CG solver with dynamic step-size control is used to solve the constrained minimization problem in (1) (in lines 10–17).

During uncoarsening, all blocks inside a cluster inherit the center position of the original cluster. *Macro shifting for LG* and *WSA for density control* are applied between uncoarsening levels. We will explain them in Sections III-A4 and A5, respectively. Then, the blocks are declustered, providing the initial placement for the next level.

To measure the evenness of the block distribution, *discrepancy* is used in [5]. They define discrepancy as the maximum ratio of the actual total block area to the maximum allowable block area over all windows within the chip. Unlike their method, we use the *overflow ratio* to measure the evenness of block distribution. We define the overflow ratio as the total overflow area in all bins over the area of total movable blocks as follows:

$$overflow\_ratio = \frac{\sum_b \max\left(D_b(\mathbf{x},\mathbf{y}) - M_b, 0\right)}{\sum \text{total movable area}} \tag{11}$$

where $overflow\_ratio \geq 0$. The overflow ratio has a more global view since it considers all overflow areas in the placement region while discrepancy only considers the maximum density of a window in the placement region. The GP stage stops when the overflow ratio is less than or equal to a user-specified target value, which is 0 by default.

Fig. 4 shows the block spreading process (Lines 10–17 of the algorithm in Fig. 3). Each time we increase the value of λ, solve the nonlinear equation, and obtain a placement result with fewer overlaps. The block spreading process continues until the total overflow ratio is small enough. Then, the spreading process stops, and all blocks are declustered into the next level.

2) CG Search With Dynamic Step Sizes: We use the CG algorithm to minimize (9). APlace uses the golden section line search to find the optimal step size, which takes most portion of its runtime during the minimization process. Instead, our step size is computed by a more efficient method. After computing the CG direction d_k, the step size α_k is computed by

$$\alpha_k = \frac{s w_b}{\|d_k\|_2} \tag{12}$$

where s is a user-specified scaling factor, and w_b is the bin width. By doing so, we can limit the step size of block spreading since the total quadratic Euclidean movement is fixed as

$$\sum_{v_i \in V} \left(\Delta x_i^2 + \Delta y_i^2\right) = \|\alpha_k d_k\|_2^2 = s^2 w_b^2 \tag{13}$$

where Δx_i and Δy_i are the amount of movement along the x- and y-directions for the block v_i in each iteration, respectively.

The value of s affects the precision of objective minimization; smaller s values lead to better results but longer runtime. In Fig. 5, the CPU times and HPWLs are plotted as functions of the step sizes. 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. In our

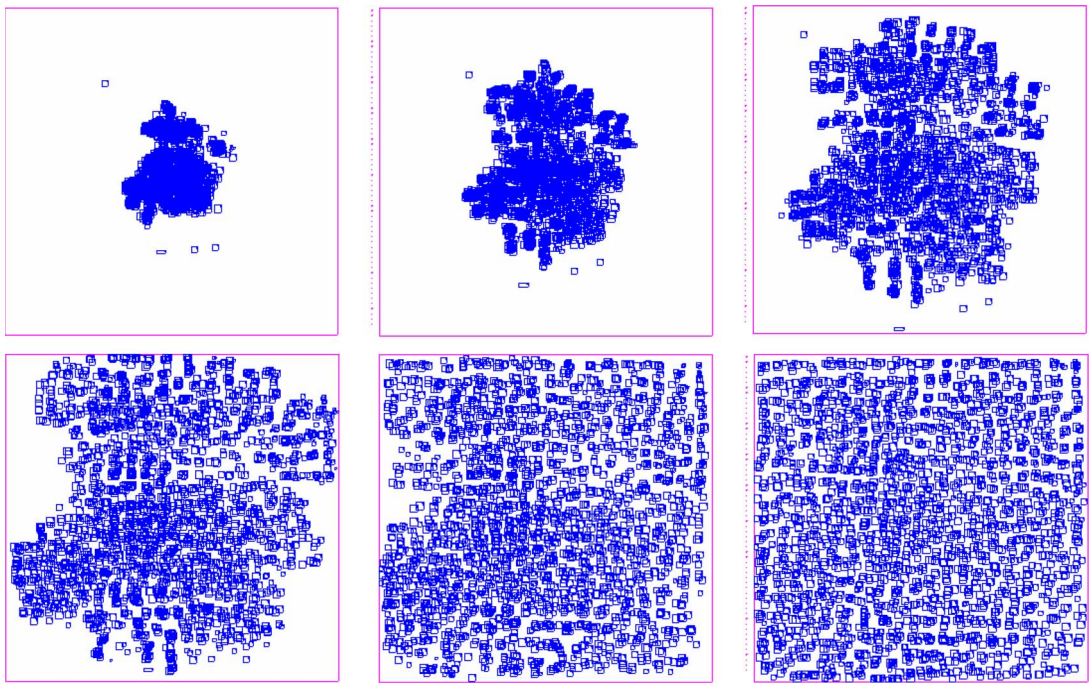

Fig. 4. Block spreading process. As the overlap weight λ increases, the overlaps are gradually reduced. The process stops when the total overflow ratio is small enough.

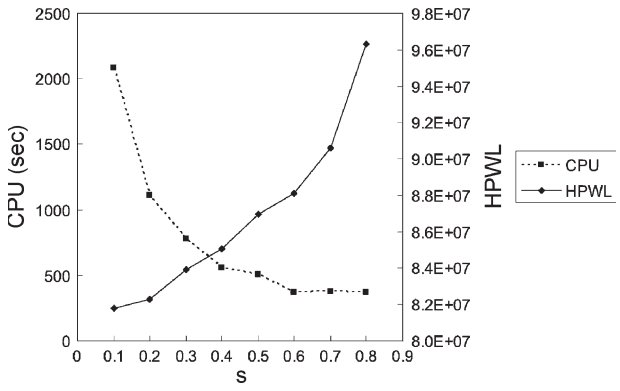

Fig. 5. CPU times and HPWLs resulting from different step sizes based on the circuit adaptec1.

Algorithm: Conjugate Gradient Algorithm with Dynamic Step-Size Control

Input:
$f(x)$: objective function
x_0: initial solution
s: step size
Output:
optimal x^*

01. initialize $g_0 = 0$ and $d_0 = 0$;
02. **do**
03. compute gradient directions $g_k = \nabla f(x_k)$;
04. compute the Polak-Ribiere parameter $\beta_k = \frac{g_k^T(g_k - g_{k-1})}{||g_{k-1}||^2}$;
05. compute the conjugate directions $d_k = -g_k + \beta_k d_{k-1}$;
06. compute the step size $\alpha_k = s/||d_k||_2$;
07. update the solution $x_k = x_{k-1} + \alpha_k d_k$;
08. **until** $(f(x_k) > f(x_{k-1}))$

Fig. 6. Our nonlinear placement objective solver. This algorithm is called in Line 11 of the multilevel GP in Fig. 3.

implementation, we set s between 0.2 and 0.3 to obtain a good tradeoff between runtime and quality.

Fig. 6 shows our CG algorithm for minimizing the placement objective during GP.

3) Base Potential Smoothing: Preplaced blocks predefine the *base potential*, which significantly affects block spreading (Fig. 7). Since the base potential P_b is not smooth, it forms 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, removing the rugged regions in the base potential, and then smooth the base potential level so that movable 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" between the adjacent regions of blocks. Fig. 8(a) shows the base potential generated by the bell-shaped function. The z-coordinate is the value of $P_b/(w_b h_b)$. If a bin has $z > 1$, it means that the potential in the bin is larger than the bin area. There are several valleys in the bottom-left regions, as shown in the figure; these regions do not have free space, but their potentials are so low that a large number of blocks may spread to these regions. To avoid this problem, we calculate the exact density as the base potential and then use the Gaussian function to smooth the base potential. The 2-D Gaussian has the form

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{14}$$

where σ is the standard deviation of the distribution. Applying convolution to the Gaussian function G with the base

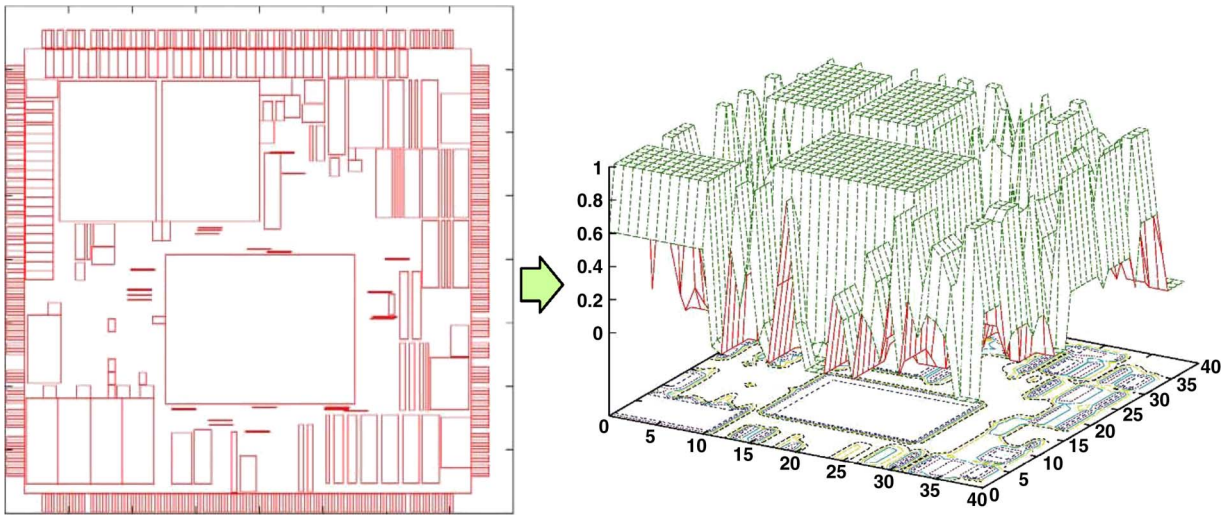

Fig. 7. Preplaced blocks and the corresponding exact base potential for the circuit adaptec2.

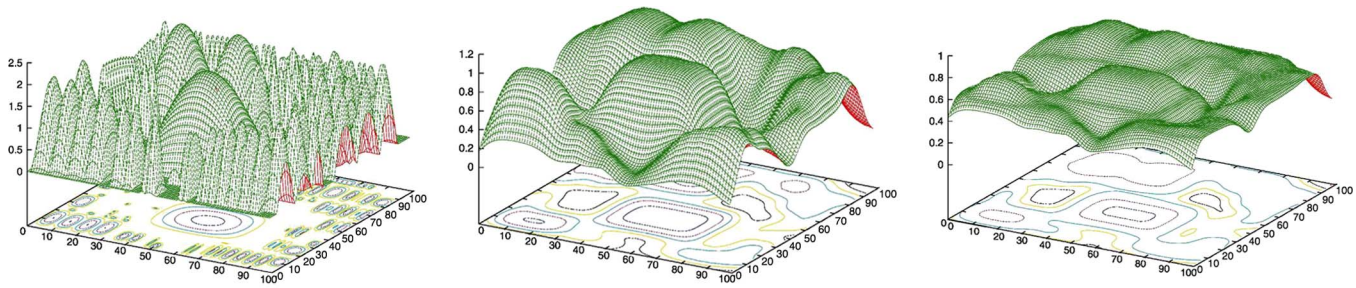

Fig. 8. Resulting base potential using different smoothing techniques. The z-coordinate is the value of $P_b/(w_b h_b)$. Note that for a region with the potential level >1.0, it means that the base potential in the region is larger than the bin area. (a) Bell-shaped smoothing. (b) Gaussian smoothing, resulting in a better smoothing potential. (c) Gaussian smoothing with level smoothing. Note that the potential level is between 0.3 and 0.8, in which blocks can more easily be spread to the whole chip.

potential P as

$$P'(x,y) = G(x,y) * P(x,y) \tag{15}$$

we can obtain a smoother base potential P'. Gaussian smoothing works as a low-pass filter, which can smooth the local density change, and the value σ defines the smoothing range. A larger σ leads to a more smooth potential. In GP, the smoothing range gradually decreases so that the smoothed potential gradually approaches the exact density. Fig. 8(b) shows the resulting potential by making σ equal to 0.25 times the chip width.

After the Gaussian smoothing, we apply another landscape smoothing function [17], [18] to reduce the potential levels. The smoothing function $P''(x,y)$ is defined as

$$P''(x,y) = \begin{cases} \overline{P'} + \left(P'(x,y) - \overline{P'}\right)^{\delta}, & \text{if } P'(x,y) \geq \overline{P'} \\ \overline{P'} - \left(\overline{P'} - P'(x,y)\right)^{\delta}, & \text{if } P'(x,y) \leq \overline{P'} \end{cases} \tag{16}$$

where $\overline{P'}$ is the average value of $P'(x,y)$, and $\delta \geq 1$. We normalize P' so that every P' is between 0 and 1 to ensure that $|P'(x,y) - \overline{P'}| < 1.0$. δ decreases from a large number (e.g., 5) to 1, and a series of level-smoothed potential is generated. Smoothing potential levels reduce "mountain" (high potential regions) heights so that movable blocks can smoothly spread to the whole placement area. Fig. 8(c) shows the resulting level-smoothed potential of Fig. 8(b) using $\delta = 2$.

4) Macro Shifting: In the GP stage, it is important to preserve legal macro positions since illegal macro positions may make the task of LG much more difficult. To avoid this, we apply macro shifting at each declustering level of the GP stage. Right after each level of uncoarsening, macro shifting first determines an order for all unclustered macros by both their coordinates and sizes (similar to our mixed-size LG described in Section III-B). Macro shifting then moves those macros to their closest legal positions by diamond search according to the predetermined order. Take the GP shown in Fig. 9(a) as an example. Two macros spread to the positions where no nearby legal positions can be found. After applying the macro shifting, we can obtain legal macro positions as shown in Fig. 9(b).

Integrating with our 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.

5) WSA for Density Control: After block spreading, some regions may still have overflows. We reduce the overflows by assigning an appropriate amount of white space. Unlike

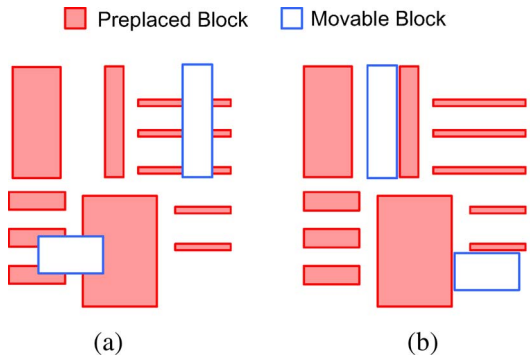

Fig. 9. Example of macro shifting. (a) A given GP with two macros being placed at illegal positions. (b) Macro shifting result with legal macro positions.

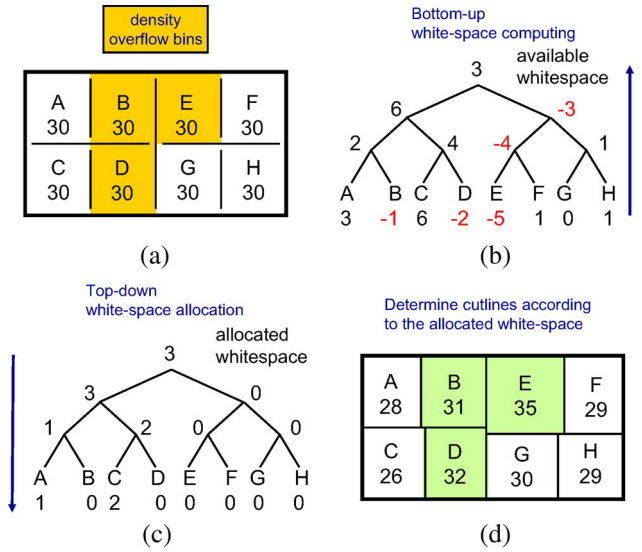

Fig. 10. (a) Initial partitions with the overflow regions being marked. (b) Corresponding slicing tree after the bottom-up white space calculation. (c) Allocated white space amount after the top-down WSA. (d) Corresponding partitions after the WSA.

the method proposed in [4] that applies WSA to reduce the routing congestion, we use WSA to remove overflow regions. We recursively partition the placement region and construct a slicing tree to record the cut directions and blocks inside the partition until the partitioned area is similar to that of a GP bin. To prevent from generating subpartitions with large aspect ratios, we choose the larger side to evenly divide the partition into two subpartitions. The process is similar to a partitioning-based GP flow, and the difference is that we divide the partition based on geometric locations of blocks instead of the cut size minimization. Fig. 10(a) shows the initial partitions and the cut lines, and each partition has an area of 30.

After the construction of the partitions and the slicing tree, we compute the white space in each partition and update the data structures for the leaf nodes of the slicing tree. A negative white space value $w < 0$ means that the partition has an overflow area of $|w|$. The partitions B, D, and E have overflow areas of 1, 2, and 5, respectively. Then, the white space of an internal node can be computed by summing up the white space of its two children. Fig. 10(b) shows the white space for every node after the bottom-up process. The white space of the root is 3, and it should always be greater than or equal to 0, or the blocks can never fit into the placement region.

After the white space calculation, the white spaces are distributed to the two children in a top-down process according to the following rules.

1) If a child node has white space $w < 0$, we allocate 0 white space to this child and allocate the remaining white space to the other child.
2) If the two children both have white spaces greater than or equal to 0, we allocate the white space proportional to their original white space amount.

The new partition area a' can be computed by $a' = a + w' - w$, where a is the old partitioned area, w is the old white space, and w' is the new white space. The cut-line adjustment is also performed in a top-down fashion. We can know the desired areas of the two subpartitions from the data structure of the two children, and then the cut line is accordingly shifted. Fig. 10(c) shows the WSA after the top-down process, and Fig. 10(d) illustrates the new partitions after the cut-line adjustment.

Finally, the new block positions can be computed by linear interpolation of the coordinates of the old partition and the new one.

B. LG

1) Mixed-Size LG: To obtain a better solution from the GP result, the LG stage removes all overlaps with minimal total displacement. We extend the standard-cell LG method in [19] to solve the mixed-size LG problem. In our LG stage, the LG orders of macros and cells are determined by their x coordinates, widths, and heights. The LG priority of a block v_i is given by

$$priority(v_i) = k_1 x_i + k_2 w_i + k_3 h_i \tag{17}$$

where k_1, k_2, and k_3 are user-specified weights for each term. (In our implementation, we use $k_1 = 1000$ and $k_2 = k_3 = 1$ by default.) Since macros have larger widths/heights, they are legalized earlier than standard cells when they have the same x coordinate. Another difference between macros and cells is that cells are packed into rows while macros are placed to their nearest available positions.

2) Look-Ahead LG: It is often hard to determine when to stop the block spreading during GP. If the blocks do not spread enough, the wirelength may significantly be increased after LG since the blocks are overcongested. If the blocks spread too much, the wirelength before LG may not be good, and even the LG step only slightly increases the wirelength. This situation becomes even worse when the density is also considered, since the placement objective is more complex. Thus, we incorporate the LG process into the GP process (see Fig. 11).

We use a look-ahead LG technique to find a desired solution. At the finest level, we apply LG after minimizing the nonlinear objective in each iteration and record the best result that has the minimum cost (wirelength and density penalty). Although look-ahead LG may take a longer runtime due to more iterations of LG, we can ensure that blocks do not overspread and thus obtain

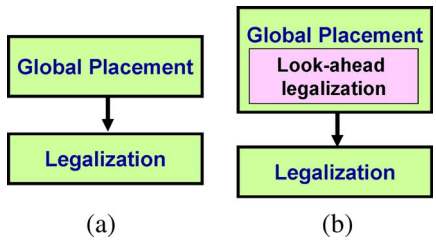

Fig. 11. (a) Traditional flow, GP followed by LG. (b) A look-ahead LG scheme during GP is used to obtain a better legal placement result.

a better legal placement. It should be noted that our look-ahead LG differs from the pre-LG schemes of PATOMA [20] and PolarBear [21] in two aspects. First, our look-ahead LG is used to reserve the desired GP result in the finest level, whereas those of PATOMA and PolarBear are used to guarantee the legality during their recursive bipartitioning process. Second, our look-ahead LG applies a Tetris-like LG to find the legalized results, whereas PATOMA and PolarBear apply the row-oriented block packing.

In our implementation, we activate the look-ahead LG only when the $overflow\_ratio$ is less than 10%. Usually, the LG is called about two to four times, and the best legalized result is reported after GP.

C. DP

In the DP stage, we try to optimize the placement result without moving macro blocks. We use the following techniques: 1) cell matching and branch-and-bound cell swapping for wirelength optimization and 2) cell sliding for density optimization.

1) Wirelength Minimization: We extend the window-based DP (WDP) algorithm [10] 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 all empty slots in the window. To keep the legality of the placement solution, for each slot, we only construct the matching relation for cells with widths less or equal to the slot width. The cost is given by the HPWL difference of a cell in each empty slot. In this paper, we implement the shortest augmenting path algorithm [22] to solve the bipartite matching problem. Though the bipartite matching problem can optimally be solved in polynomial time, the optimal assignment cannot guarantee the optimal HPWL result because the HPWL cost of a cell connected to each empty slot depends on the positions of 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 pair of the selected cells. We also observed that the bipartite matching problem can very quickly be solved when the problem size is smaller than 100 cells. Therefore, in addition to the cells selected from a local window, we also randomly select cells from the full placement region in each run of the cell matching. Compared with other DP algorithms, cell matching can more globally optimize the placement result.

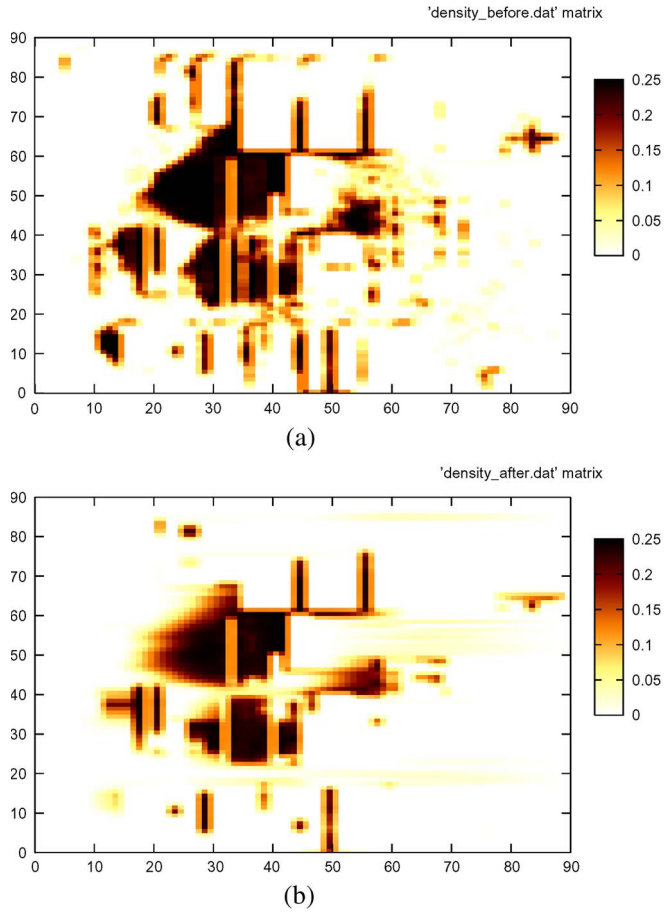

Fig. 12. Density map for the circuit adaptec1. Dark color represents the density overflow, and the value is defined by $\max(0, D_b/M_b - 1)$. The target density t_{density} is set to 0.8. The overflow ratio is 0.022% (0.015%) before (after) cell sliding. The *density penalty* (defined in Section IV-C) is reduced from 6.32% to 3.09%, whereas HPWL only increases by 0.85%. (a) Before cell sliding. (b) After cell sliding.

2) Density Optimization: In addition to wirelength minimization during the DP, we optimize the chip density by cell sliding. The objective of density optimization is to reduce the density overflow in the congested area. In this stage, all the macro blocks are fixed, and we only consider standard cells. We divide the placement region into uniform bins, and then our algorithm iteratively reduces the densities of overflowed bins by sliding the cells from denser bins to sparser ones while the cell order is preserved. Since vertical sliding often generates misalignment between standard cells and site rows for row-based designs, we only implement horizontal sliding to maintain the legality of the placement solution. Each iteration consists of two phases, i.e., left sliding and right sliding. In each phase, 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'. $f_{bb'}$ denotes the desired amount of movable cell area to move from bin b to bin b'. Recall that we define D_b as the total area of the movable cells in bin b, and M_b as the maximum allowable area of movable blocks in bin b. If bin b does not have any 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

TABLE II
ICCAD'04 IBM MIXED-SIZE BENCHMARK STATISTICS

Circuit	#Cells	#Macros	#Pads	#Nets	#Util (%)
ibm01	12260	246	246	14111	80.02
ibm02	19071	271	259	19584	80.02
ibm03	22563	290	283	27401	80.01
ibm04	26925	295	287	31970	80.02
ibm05	28146	0	1201	28446	80.01
ibm06	32154	178	166	34826	80.00
ibm07	45348	291	287	48117	80.01
ibm08	50722	301	286	50513	80.00
ibm09	52857	253	285	60902	80.00
ibm10	67899	786	744	75196	80.01
ibm11	69779	373	406	81454	80.00
ibm12	69788	651	637	77240	80.01
ibm13	83285	424	490	99666	80.02
ibm14	146474	614	517	152772	80.01
ibm15	160794	393	383	186608	80.00
ibm16	182522	458	504	190048	80.01
ibm17	183992	760	743	189581	80.01
ibm18	210056	285	272	201920	80.01

TABLE III
ISPD'05 PLACEMENT CONTEST BENCHMARK STATISTICS

	#Movs	#Fixed	#Nets	Util (%)
adaptec1	211k	543	221k	57.34
adaptec2	254k	566	266k	44.32
adaptec3	451k	723	467k	33.52
adaptec4	495k	1329	516k	27.14
bigblue1	278k	560	284k	44.67
bigblue2	535k	23084	577k	37.78
bigblue3	1096k	1293	1123k	56.48
bigblue4	2169k	8170	2230k	44.29

to bin b'. Otherwise, we evenly distribute the overflow area between b and b'. Therefore, $f_{bb'}$ is defined by

$$f_{bb'} = \begin{cases} D_b - M_b, & \text{if } (M_{b'} - D_{b'}) \geq (D_b - M_b) \\ \frac{D_b M_{b'} - D_{b'} M_b}{M_b + M_{b'}}, & \text{otherwise} \end{cases} \tag{18}$$

where the second condition is derived from

$$D_b - \left(M_b + \frac{(D_b - M_b + D_{b'} - M_{b'}) M_b}{M_b + M_{b'}} \right) = \frac{D_b M_{b'} - D_{b'} M_b}{M_b + M_{b'}}. \tag{19}$$

After the area flow $f_{bb'}$ is computed, the area flow is then evenly distributed to all rows within the overflow bin. If the area flow of a row is larger than the area of the leftmost (rightmost) cell, we keep the concatenating cell on its right (left) until the leftmost (rightmost) cell is large enough. Then, we can obtain the coordinates of the leftmost (rightmost) cells that satisfy the area flow of each row. If one row fails to slide out the required area flow, the insufficient amount will again evenly be distributed to the other rows, and the sliding process repeats until the required area flow is reached or no further movement is possible. Then, we update D_b and $D_{b'}$. In the right sliding phase, we start from the leftmost bin of the placement region, and b' is right to b. In the left sliding phase, we start from the rightmost bin, and b' is accordingly left to b. We iteratively slide the cells from the area overflow region to a sparser region until no significant improvement can be obtained. Fig. 12 shows the density map before and after our cell-sliding procedure on the circuit adaptec1.

TABLE IV
STATISTICS FOR THE ISPD'06 PLACEMENT CONTEST BENCHMARKS

	#Movs	#Fixed	#Nets	Util (%)	Density Target (%)
adaptec5	842k	646	868k	49.98	50
newblue1	330k	337	339k	70.79	80
newblue2	330k	1277	465k	61.66	90
newblue3	483k	11178	552k	26.31	80
newblue4	643k	3422	637k	46.45	50
newblue5	1228k	4881	1284k	49.56	50
newblue6	1248k	6889	1288k	38.78	80
newblue7	2481k	26582	2637k	49.31	80

TABLE V
COMPARISON AMONG OUR PLACER (NTUplace3), APLACE 2.0, AND mPL6 ON THE ICCAD'04 IBM MIXED-SIZE BENCHMARKS

	NTUplace3		APlace 2.0		mPL6	
	HPWL (× e6)	CPU (sec)	HPWL (× e6)	CPU (sec)	HPWL (× e6)	CPU (sec)
ibm01	2.17	30	2.14	346	2.24	123
ibm02	4.63	57	4.65	793	4.76	185
ibm03	6.65	65	6.71	923	6.78	186
ibm04	7.21	81	7.57	888	7.31	216
ibm05	9.66	145	9.69	696	9.41	237
ibm06	5.94	86	6.02	879	5.81	257
ibm07	9.90	199	10.00	1178	9.79	363
ibm08	12.29	214	12.50	1349	11.31	544
ibm09	12.00	194	12.13	1670	12.47	582
ibm10	28.49	319	28.83	2408	27.83	734
ibm11	17.54	305	18.67	3467	17.79	692
ibm12	32.07	302	33.42	3330	31.52	804
ibm13	22.16	487	22.80	3495	22.97	917
ibm14	35.36	1158	35.92	4294	36.53	1463
ibm15	45.38	1337	46.81	4926	47.44	1825
ibm16	57.59	1450	54.53	5554	55.99	2688
ibm17	66.73	1930	65.67	6032	66.16	1943
ibm18	41.58	2613	41.99	9932	43.37	2307
average	1.00	1.00	1.01	7.87	1.01	2.16

TABLE VI
COMPARISON AMONG OUR PLACER (NTUplace3), APLACE 2.0, AND mPL6 ON THE ISPD'05 PLACEMENT CONTEST BENCHMARKS

	NTUplace3		APlace 2.0		mPL6	
	HPWL (× e6)	CPU (sec)	HPWL (× e6)	CPU (sec)	HPWL (× e6)	CPU (sec)
adaptec1	80.93	803	78.35	7001	77.85	2019
adaptec2	89.85	824	95.70	9793	90.99	2308
adaptec3	214.20	1767	218.52	24849	215.87	6666
adaptec4	193.74	2114	209.28	29377	193.94	6619
bigblue1	97.28	1523	100.01	10318	97.27	2512
bigblue2	152.20	3047	153.75	24789	151.76	7521
bigblue3	348.48	5687	411.59	44805	345.11	10447
bigblue4	829.16	10280	871.29	115363	830.30	23492
average	1.00	1.00	1.05	10.32	1.00	2.56

IV. EXPERIMENTAL RESULTS

We compared our placer with APlace 2.0 and mPL6, which achieved the best published results among all publicly available placers, based on the 2004 International Conference on Computer Aided Design (ICCAD'04) IBM mixed-size [23] and the 2005 International Symposium on Physical Design (ISPD'05) placement contest [24] benchmark suites. The statistics are shown in Tables II–IV, respectively. Note that APlace 2.0

TABLE VII
HPWL (× E6) COMPARISON BASED ON THE ISPD'06 BENCHMARKS

HPWL	NTUplace3	APlace3 [13]	Capo [27]	DPlace	Dragon [28]	FastPlace	Kraftwerk [29]	mFAR	mPL6 [6]	NTUplace2 [10]
adaptec5	378.56	449.61	491.60	463.95	500.24	478.47	444.07	448.43	425.12	404.98
newblue1	60.74	73.26	98.35	102.37	80.76	84.49	78.29	77.36	66.90	62.40
newblue2	198.76	197.42	308.64	324.07	259.95	209.73	205.87	211.65	197.53	201.95
newblue3	278.87	273.63	361.21	379.19	524.41	361.05	279.94	303.58	283.80	291.14
newblue4	274.48	377.55	358.28	305.78	340.70	319.08	311.09	307.73	294.43	284.99
newblue5	474.84	545.90	657.40	600.11	613.34	601.45	555.48	567.65	530.67	494.57
newblue6	484.81	522.58	668.33	674.39	572.19	539.16	537.32	527.36	510.40	504.39
newblue7	1056.78	1098.26	1518.49	1398.85	1408.97	1173.15	1139.17	1135.80	1070.33	1116.86
average	1.00	1.13	1.41	1.37	1.36	1.21	1.12	1.14	1.06	1.04

TABLE VIII
COMBINED DENSITY AND HPWL (DHPWL) (× E6) COMPARISON BASED ON THE ISPD'06 BENCHMARKS

DHPWL	NTUplace3	APlace3 [13]	Capo [27]	DPlace	Dragon [28]	FastPlace	Kraftwerk [29]	mFAR	mPL6 [6]	NTUplace2 [10]
adaptec5	448.58	520.97	494.64	572.98	500.74	805.63	457.92	476.28	431.14	432.58
newblue1	61.08	73.31	98.48	102.75	80.77	84.55	78.60	77.54	67.02	63.49
newblue2	203.39	198.24	309.53	329.92	260.83	212.30	208.41	212.90	200.93	203.68
newblue3	278.89	273.64	361.25	380.14	524.58	362.99	280.93	303.91	287.05	291.15
newblue4	301.19	384.12	362.40	364.45	341.16	429.78	315.53	324.40	299.66	305.79
newblue5	509.54	613.86	659.57	752.07	614.23	962.06	569.36	601.27	540.67	517.63
newblue6	521.65	522.73	668.66	682.87	572.53	574.18	545.94	535.96	518.70	532.79
newblue7	1099.66	1098.88	1518.75	1438.99	1410.54	1236.34	1170.85	1153.76	1082.91	1181.30
average	1.00	1.10	1.34	1.41	1.29	1.38	1.08	1.10	1.01	1.02

and mPL6 are the latest publicly available versions. All results were generated on the same PC workstation with an Opteron 2.4-GHz CPU based on the default parameters given in each placer, and no manual parameter tuning for individual circuits is allowed for fair comparison. Note that we tested on APlace 2.0 based on its default mode instead of taking the results of APlace 2.0 reported in [12] since manual parameter tuning to each circuit of the ISPD'05 placement contest benchmark suite was applied to obtain those results, and it needs much longer CPU times than those with the default mode.

We also compared with other eight state-of-the-art academic placers, such as APlace 3.0 and mPL6, based on the ISPD'06 placement contest benchmark suite [25]. Since the eight academic placers are not available to us, we reported the results given in [25] and [26].

A. ICCAD'04 IBM Mixed-Size Benchmarks

In the first experiment, we evaluated the performance of our placer on the ICCAD'04 IBM mixed-size benchmark suite. These benchmarks have nontrivial macro aspect ratios and pin locations for individual cells or macros, and thus are more realistic to modern circuit designs. Table V lists the HPWLs and CPU times for our placer, APlace 2.0, and mPL6. The last row in Table V shows the average normalized wirelength and CPU time ratio based on our results. Compared with APlace 2.0, our placer achieves 1% shorter wirelength and is 7.87× faster. Compared with mPL6, our placer obtains 1% shorter wirelength and is 2.16× faster. On average, our placer produces the best solution quality in a smaller runtime.

B. ISPD'05 Placement Contest Benchmarks

The ISPD'05 benchmarks have circuit sizes (placeable blocks) ranging from 211 to 2169 K, and the physical structure of these designs is completely preserved. These benchmarks contain a large amount of white space, fixed blocks, and I/Os, and give realistic challenges for modern placers. Table VI lists the results of ours, APlace 2.0, and mPL6. As shown in the table, our placer achieves the best average wirelength in the shortest CPU time. On average, our resulting HPWL is smaller than that of APlace 2.0 by 5% and similar to mPL6's, and our placer is 10.32× and 2.56× faster than APlace 2.0 and mPL6, respectively.

C. ISPD'06 Placement Contest Benchmarks

In the third experiment, we reported the results on the ISPD'06 placement contest benchmark suite [25]. The results of other placers were taken from [25] and [26]. Compared to ISPD'05 benchmarks, the new benchmark suite has more movable blocks and wider ranges of design utilizations. Tables VII–IX compare the HPWL, density HPWL (DHPWL), and CPU time of the placers on the ISPD'06 benchmarks, respectively. The target density t_{density} of each circuit is set according to the number given in the last column of Table IV, and the CPU times were measured on an Opteron 2.6-GHz PC machine. The DHPWL is defined as [25], [26]

$$\text{DHPWL} = \text{HPWL} \times (1 + density\_penalty). \quad (20)$$

To compute $density\_penalty$, we made the bin grid width and height equal to ten circuit row height, and $density\_penalty$ is defined by

$$density\_penalty = (overflow\_ratio \times bin\_area \times density\_target)^2 \quad (21)$$

and $overflow\_ratio$ is defined by (11).

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 [25], [26],

TABLE IX
CPU TIME (IN SECONDS) COMPARISON BASED ON THE ISPD'06 BENCHMARKS. OUR CPU TIME (NTUplace3) IS MEASURED ON AN OPTERON 2.4-GHz MACHINE, WHILE OTHERS ARE ON AN OPTERON 2.6-GHz MACHINE

CPU Time	NTUplace3	APlace3 [13]	Capo [27]	DPlace	Dragon [28]	FastPlace	Kraftwerk [29]	mFAR	mPL6 [6]	NTUplace2 [10]
adaptec5	4718	20267	9718	2877	2257	4055	3293	6875	8265	10494
newblue1	1168	4303	2562	1026	989	516	1135	2538	2252	2163
newblue2	2750	5533	5642	6393	1631	1033	1007	2892	6089	4425
newblue3	1670	12503	6076	1028	1170	2437	912	2958	9696	6646
newblue4	3627	14982	6926	1647	1486	1388	2772	6362	5815	7468
newblue5	17955	32799	20854	4550	3529	6224	7423	11426	12349	20441
newblue6	9679	29124	18485	4032	3860	4157	5350	12154	12035	13849
newblue7	20436	54852	54962	9508	9902	6624	7465	19484	28385	21464
average	1.00	3.64	2.20	0.75	0.51	0.58	0.59	1.38	2.08	1.92

TABLE X
HPWL AND RUNTIME RESULTS FOR THE ISPD'05 BENCHMARK SUITE

Circuit	HPWL (× e6)			CPU (sec)			
	GP	LG	DP	GP	LG	DP	Total
adaptec1	82.10	83.92	80.93	608	105	80	803
adaptec2	90.39	93.07	89.85	640	36	135	824
adaptec3	207.09	227.12	214.20	1308	199	237	1767
adaptec4	196.63	203.28	193.74	1637	150	303	2114
bigblue1	92.19	100.54	97.28	1268	139	103	1523
bigblue2	152.24	161.04	152.20	2414	167	439	3047
bigblue3	331.35	382.72	348.48	2793	2122	723	5687
bigblue4	812.62	884.10	829.16	6127	2089	1957	10280
ratio	1.00	1.07	1.02	72%	14%	13%	100%

TABLE XI
HPWL AND RUNTIME RESULTS FOR THE ISPD'06 BENCHMARK SUITE

Circuit	HPWL (× e6)			DHPWL	CPU (sec)			
	GP	LG	DP	(× e6)	GP	LG	DP	Total
adaptec5	373.84	402.38	375.05	448.58	8366	821	700	9971
newblue1	59.68	62.41	60.36	68.10	821	89	262	1194
newblue2	186.33	213.41	198.63	203.39	1961	1178	198	3380
newblue3	291.03	295.98	278.87	278.89	1107	53	690	1883
newblue4	274.52	287.98	271.01	301.19	5775	388	595	6812
newblue5	511.67	511.67	469.95	509.54	15601	1044	1159	17899
newblue6	474.83	511.02	482.19	521.65	13558	459	1105	15426
newblue7	1037.43	1126.88	1051.13	1099.66	23464	3006	1860	28734
average	1.00	1.07	1.01	1.08	79%	10%	11%	100%

placers with 2× (4×) 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, mPL6, and NTUplace2, respectively.

D. HPWL and Runtime Analysis

Table X lists the HPWLs and CPU times of the GP, LG, and DP stages for the ISPD'05 benchmark suite. On average, the LG stage increases the wirelength by 7%, whereas the DP stage decreases the wirelength by 5%. For the CPU time, GP spends 72% of the total runtime, which is much more than those of the LG and DP stages.

Table XI shows the HPWL, DHPWL (combined cost with wirelength and density), and CPU time of every placement stage for the ISPD'06 benchmark suite. Similar to the results for the ISPD'05 benchmark suite, on average, the LG stage increases the wirelength by 7%, and the DP stage decreases the wirelength by 6%. It should be noted that the DP result only incurs 7% density penalty. Again, most of the CPU time was spent on GP (79%).

TABLE XII
NORMALIZED DHPWLs WITH SOME INDIVIDUAL TECHNIQUE BEING TURNED OFF FOR THE ISPD'06 BENCHMARK SUITE

Circuit	GP		LG	DP	
	w/o LAL	w/o Macro Shifting	Plain Legalization	w/o Cell Matching	w/o Cell Sliding
adaptec5	1.04	1.00	1.00	1.03	1.02
newblue1	1.08	1.03	1.00	1.01	1.01
newblue2	1.03	1.00	1.00	1.04	1.01
newblue3	1.00	1.00	1.00	1.01	1.00
newblue4	1.10	1.00	1.00	1.02	1.02
newblue5	1.10	1.00	1.00	1.02	1.01
newblue6	1.01	1.00	1.00	1.02	1.01
newblue7	1.01	1.00	1.00	1.03	1.01
average	1.05	1.00	1.00	1.02	1.01

E. Effects of Individual Techniques

In this experiment, we analyzed the effects of individual techniques applied in NTUplace3. Table XII summarizes the resulting DHPWL ratios, which are normalized to the DHPWLs of the complete NTUplace3, with some individual technique being turned off. The columns "w/o LAL" and "w/o Macro Shifting" give the results with look-ahead LG and macro shifting being turned off, respectively. The column "Plain LG" represents that the LG order is determined by the x coordinate

TABLE XIII
HPWL IMPACTS OF MACRO SHIFTING ON THE ICCAD'04 IBM MIXED-SIZE BENCHMARKS

Circuit	w/ Macro Shifting	w/o Macro Shifting	Circuit	w/ Macro Shifting	w/o Macro Shifting
ibm01	2.17	2.17	ibm10	28.49	31.63
ibm02	4.63	5.22	ibm11	17.54	17.47
ibm03	6.65	6.73	ibm12	32.07	33.25
ibm04	7.21	7.57	ibm13	22.16	22.45
ibm05	9.66	9.31	ibm14	35.36	34.71
ibm06	5.94	5.89	ibm15	45.38	47.04
ibm07	9.90	9.93	ibm16	57.59	59.61
ibm08	12.29	12.96	ibm17	66.73	66.86
ibm09	12.00	11.96	ibm18	41.58	42.56
average	-	-	-	1.00	1.02

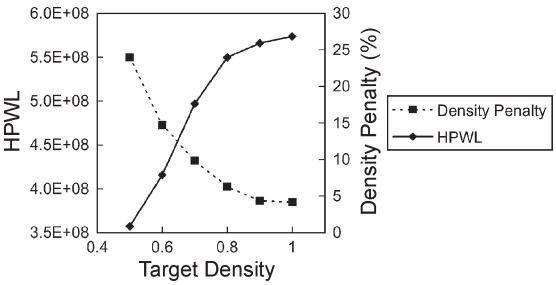

Fig. 13. HPWLs and density penalties resulting from different WSA target densities based on the circuit adaptec5.

of each block alone, whereas our LG gives higher priorities to macros. The columns "w/o Cell Matching" and "w/o Cell Sliding" give the results with cell matching and cell sliding being turned off, respectively. As shown in the table, since look-ahead LG preserves the desired placement result in the finest level, NTUplace3 achieves 5% better average DHPWL than that without the look-ahead LG. Further, because the macro shifting provides better macro positions for LG, NTUplace3 obtains 3% better DHPWL in newblue1, which contains a number of movable large macros. In the LG stage, NTUplace3 generates comparable results with that of plain LG, implying that allowing macros to have a higher priority does not harm the placement quality. Further, if the design utilization is high, there are fewer spaces for macro LG; legalizing macros earlier would increase the success rate of the LG. NTUplace3 can get respective 2% and 1% better average DHPWLs than those without cell matching and cell sliding, implying that the two DP techniques are effective in reducing the HPWL and the density penalty.

To further demonstrate the impacts of macro shifting on circuits with many movable macro blocks, we also tested NTUplace3 with macro shifting turned off on the ICCAD'04 IBM mixed-size benchmark suite. The HPWL results are summarized in Table XIII. As shown in the table, NTUplace3 with macro shifting on average can achieve 2% shorter HPWL than that without macro shifting, implying again that the macro shifting can provide better macro positions with the existence of many macro blocks.

Fig. 13 shows the effect of WSA on the circuit adaptec5. Although not presented here, the trend for other circuits are similar. The target density for WSA affects the tradeoff between HPWL and density penalty. When the target density for WSA is 0.5, we can obtain a placement with only 0.84% penalty. When the target density for WSA increases, the density penalty increases and HPWL decreases. Thus, the target density for WSA can be chosen with design requirements. For DHPWL optimization, we could run WSA under several target densities and choose the best one.

V. CONCLUSION

In this paper, we have proposed a high-quality mixed-size analytical placer considering preplaced blocks and density constraints. Experimental results have shown that our placer achieves very-high-quality placement results and is very efficient.

REFERENCES

[1] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, and Y.-W. Chang, "A high-quality mixed-size analytical placer considering preplaced blocks and density constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2006, pp. 187–192.

[2] W. C. Naylor, R. Donelly, and L. Sha, "Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer," U.S. Patent 6 301 693, Oct. 9, 2001.

[3] X. Yang, B.-K. Choi, and M. Sarrafzadeh, "Routability-driven white space allocation for fixed-die standard-cell placement," in *Proc. ACM Int. Symp. Phys. Des.*, 2002, pp. 42–47.

[4] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden, "Routability-driven placement and white space allocation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2004, pp. 394–401.

[5] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 5, pp. 734–747, May 2005.

[6] T. Chan, J. Cong, J. Shinnerl, K. Sze, and M. Xie, "mPL6: Enhanced multilevel mixed-size placement," in *Proc. ACM Int. Symp. Phys. Des.*, 2006, pp. 212–214.

[7] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov, "Unification of partitioning, placement and floorplanning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2004, pp. 550–557.

[8] A. R. Agnihotri, S. Ono, and P. H. Madden, "Recursive bisection placement: Feng Shui 5.0 implementation details," in *Proc. ACM Int. Symp. Phys. Des.*, 2005, pp. 230–232.

[9] C.-C. Chang, J. Cong, and X. Yuan, "Multi-level placement for large-scale mixed-size IC designs," in *Proc. ASP-DAC*, 2003, pp. 325–330.

[10] 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. ACM Int. Symp. Phys. Des.*, 2006, pp. 215–217.

[11] B. Yao, H. Chen, C.-K. Cheng, N.-C. Chou, L.-T. Liu, and P. Suaris, "Unified quadratic programming approach for mixed mode placement," in *Proc. ACM Int. Symp. Phys. Des.*, 2005, pp. 193–199.

[12] A. B. Kahng, S. Reda, and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2005, pp. 890–897.

[13] A. B. Kahng and Q. Wang, "A faster implementation of APlace," in *Proc. ACM Int. Symp. Phys. Des.*, 2006, pp. 218–220.

[14] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proc. ACM Int. Symp. Phys. Des.*, Apr. 2005, pp. 185–192. best paper award at ISPD'2005. [Online]. Available: http://www.gigascale.org/pubs/600.html

[15] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. ACM/IEEE Des. Autom. Conf.*, 1998, pp. 269–274.

[16] M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 3, pp. 356–365, Mar. 1991.

[17] J. Gu and X. Huang, "Efficient local search with search space smoothing: A case study of the traveling salesman problem (TSP)," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 5, pp. 728–735, May 1994.

[18] A. B. Kahng, S. Reda, and Q. Wang, "APlace: A general analytic placement framework," in *Proc. ACM Int. Symp. Phys. Des.*, 2005, pp. 233–235.

[19] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design," U.S. Patent 6 370 673, Apr. 9, 2002.

[20] J. Cong, M. Romesis, and J. R. Shinnerl, "Fast floorplanning by look-ahead enabled recursive bipartitioning," in *Proc. ASP-DAC*, 2005, pp. 1119–1122.
[21] J. Cong, M. Romesis, and J. R. Shinnerl, "Robust mixed-size placement under tight white-space constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2005, pp. 165–172.
[22] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, Mar. 1987.
[23] *ICCAD04 Mixed-Size Placement Benchmarks*. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/ICCAD04bench/
[24] *ISPD 2005 Placement Contest*. [Online]. Available: http://www.sigda.org/ispd2005/contest.htm
[25] *ISPD 2006 Program*. [Online]. Available: http://www.ispd.cc/program.html
[26] G.-J. Nam, C. J. Aplert, and P. G. Villarrubia, "ISPD 2006 placement contest: Benchmark suite and results," in *Proc. ISPD*, 2006, p. 167.
[27] J. Roy, D. Papa, A. Ng, and I. Markov, "Satisfying whitespace requirements in top-down placement," in *Proc. ACM Int. Symp. Phys. Des.*, 2006, pp. 206–208.
[28] T. Taghavi, X. Yang, B.-K. Choi, M. Wang, and M. Sarrafzadeh, "Dragon2006: Blockage-aware congestion-controlling mixed-size placer," in *Proc. ACM Int. Symp. Phys. Des.*, 2006, pp. 209–211.
[29] P. Spindler and F. M. Johannes, "Fast and robust quadratic placement combined with an exact linear net model," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2006, pp. 179–186.

Tung-Chieh Chen (S'04) received the B.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2003. He is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University.

In 2007, he was a Visiting Scholar with the University of Texas, Austin. He is also currently a Senior Engineer with SpringSoft, Inc., Hsinchu, Taiwan.

Dr. Chen was the recipient of the First Prize of the ACM/SIGDA CADathlon Programming Contest in 2007.

Zhe-Wei Jiang (S'05) received the B.S. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 2003. He is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.

His current research interests focus on large-scale mixed-size placement and design for manufacturability.

Tien-Chang Hsu received the B.S. degree in electrical engineering and the M.S. degree in electronics engineering from National Taiwan University Taipei, Taiwan, R.O.C., in 2004 and 2006, respectively.

He is currently with Synopsys Taiwan Ltd., Taipei, Taiwan, R.O.C. His current research interests include large-scale mixed-size placement, routing, and design for manufacturability.

Hsin-Chen Chen (S'05) received the B.S. degree in electrical engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C., in 2005, and the M.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2007.

He is currently doing his military service. His research interests focus on floorplanning and large-scale mixed-size placement.

Yao-Wen Chang (S'94–A'96–M'96) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1988, and the M.S. and Ph.D. degrees from the University of Texas at Austin in 1993 and 1996, respectively, all in computer science.

He is a Professor in the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University. He is currently also a Visiting Professor at Waseda University, Kitakyushu, Japan. He was with the IBM T. J. Watson Research Center, Yorktown Heights, NY, in the summer of 1994. From 1996 to 2001, he was on the faculty of National Chiao Tung University, Taiwan. His current research interests lie in VLSI physical design, design for manufacturability/reliability, and design automation for biochips. He has been working closely with industry on projects in these areas. He has coauthored one book on routing and over 130 ACM/IEEE conference/journal papers in these areas.

Dr. Chang is a winner of the 2008 ACM ISPD Global Routing Contest and the 2006 ACM ISPD Placement Contest. He received Best Paper Awards at ICCD-95, and eleven Best Paper Award Nominations from DAC (four times), ICCAD (twice), ISPD (twice), ACM TODAES, ASP-DAC, and ICCD in the past eight years. He has received many awards for research performance, such as the 2007 Distinguished Research Award, the inaugural 2005 First-Class Principal Investigator Award, and the 2004 Dr. Wu Ta You Memorial Award, all from National Science Council of Taiwan, the 2004 MXIC Young Chair Professorship from the MXIC Corp, and for excellent teaching from National Taiwan University (four times) and National Chiao Tung University. He is currently an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (TCAD) and an Editor of the *Journal of Information Science and Engineering (JISE)*. He has served on the ICCAD Executive Committee, the ACM/SIGDA Physical Design Technical Committee, the ACM ISPD Organizing Committee, and the technical program committees of ASP-DAC (topic chair), DAC, DATE, FPL, FPT (program co-chair), GLSVLSI, ICCAD, ICCD, IECON (topic chair), ISPD, SOCC (topic chair), TENCON, and VLSI-DAT (topic co-chair). He is currently an independent board director of Genesys Logic, Inc, a member of board of governors of Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.