TSV-Aware Analytical Placement for 3-D IC Designs Based on a Novel Weighted-Average Wirelength Model

Meng-Kai Hsu, Valeriy Balabanov, and Yao-Wen Chang, *Fellow, IEEE*

***Abstract*—Through-silicon vias (TSVs) are required for transmitting signals among different dies for the 3-D integrated circuit (IC) technology. The significant silicon areas occupied by TSVs bring critical challenges for 3-D IC placement. Unlike most published 3-D placement works that only minimize the number of TSVs during placement due to the limitations in their techniques, this paper proposes a new 3-D cell placement algorithm that can additionally consider the sizes of TSVs and the physical positions for TSV insertion during placement. The algorithm consists of three stages: 1) 3-D analytical global placement with density optimization and whitespace reservation for TSVs; 2) TSV insertion and TSV-aware legalization; and 3) layer-by-layer detailed placement. In particular, the global placement is based on a novel weighted-average (WA) wirelength model, giving the first published model that can outperform the well-known log-sum-exp wirelength model theoretically and empirically. Also, a scheme is proposed to enhance the numerical stability of the WA wirelength model. Furthermore, 3-D routing can easily be accomplished by traditional 2-D routers since the physical positions of TSVs are determined during placement. Experimental results show the effectiveness of our algorithm. Compared with state-of-the-art 3-D cell placement works, our algorithm can achieve the best routed wirelength, TSV counts, and total silicon area, in shortest running time.**

***Index Terms*—3-D integrated circuits (ICs), layout, physical design, placement, wirelength.**

I. Introduction

THE 3-D integrated circuit (IC) technology has emerged as one of the most promising solutions for overcoming the challenges in interconnect and integration complexity in modern and next-generation circuit designs. The 3-D IC technology can effectively reduce global interconnect length and increase circuit performance; however, this technology brings some challenges with through-silicon vias (TSVs), used to make interconnections among different layers, thermal effects, packaging, power delivery/density, etc.

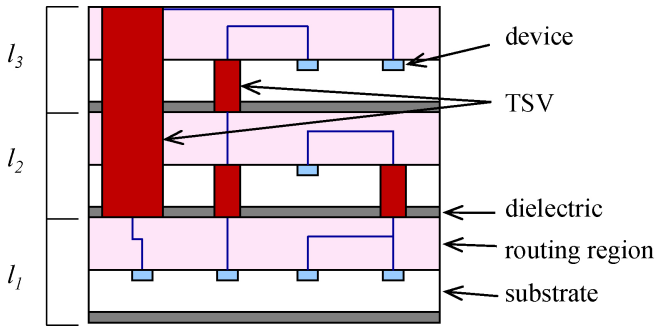


Fig. 1. Cross section view of a 3-D IC structure.

The advantages of 3-D ICs can be summarized as follows [2]: 1) higher integration density: it can place more elements into one single package using a much smaller area than a traditional 2-D IC; 2) heterogeneous integration: it can integrate disparate technologies, such as logic circuit, memory, and mixed signal components; 3) higher performance: it can significantly reduce the wirelength; and 4) lower power: it can reduce power consumption, especially for the clock network because of shorter wirelength. As a result, the 3-D ICs become a promising alternative to current 2-D ICs.

Fig. 1 shows the cross section view of a popular 3-D IC structure. As shown in the figure, each die is stacked on top of another and communicated by TSVs [3]–[8] (i.e., face-to-back stacking). These TSVs are responsible for the interconnections among devices on different layers, but they could cause some significant problems. Under current technologies, TSV pitches are very large compared to the sizes of regular metal wires; as a result, a large number of TSVs will consume significant silicon areas and degrade the yield and reliability of the final chip. Furthermore, TSVs are usually placed at the whitespace among macro blocks or cells, thus TSVs might affect the routing resource and increase the overall chip or package area. The significant silicon areas occupied by TSVs and the induced yield and reliability issues become critical problems for 3-D IC placement.

A. Previous Work

The 3-D IC placement problem has attracted increasing attention in the recent literature. By reusing modern 2-D

Manuscript received March 15, 2012; revised August 2, 2012; accepted September 29, 2012. Date of current version March 15, 2013. This work was supported in part by IBM, SpringSoft, TSMC, Academia Sinica, and NSC of Taiwan, under Grant NSC 101-2221-E-002-191-MY3, Grant NSC100-2221-E-002-088-MY3, Grant NSC 99-2221-E-002-207-MY3, and Grant NSC 99-2221-E-002-210-MY3. A preliminary version of this paper was presented at the 2011 ACM/IEEE Design Automation Conference in June 2011 [1]. This paper was recommended by Associate Editor Y. Xie.

The authors are with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: kaie@eda.ee.ntu.edu.tw; balabasik@gmail.com; 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.2012.2226584

0278-0070/$31.00 © 2013 IEEE

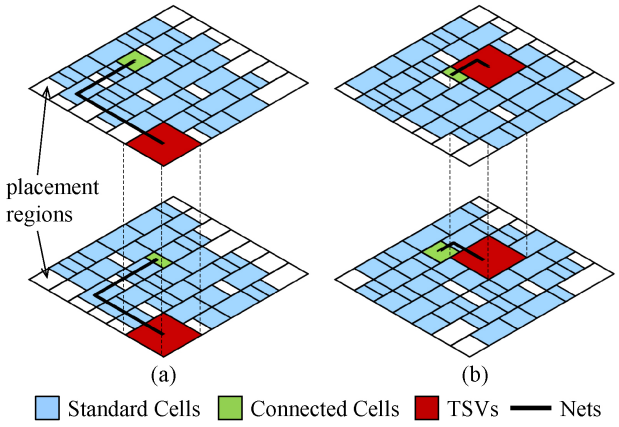


Fig. 2. Whitespace reservation for TSVs during placement. (a) Without whitespace reservation for TSVs during placement, the available whitespace for TSV insertion tends to be located along the chip periphery, especially for analytical placement; consequently, the resulting placement would incur longer wirelength since the available TSV might be far from its connected cells. (b) With whitespace reservation for TSVs, the resulting total wirelength can be much smaller.

placement results, a folding/stacking based 3-D placement method was proposed [5]. This method performs layer reassignment for cells to further improve 3-D placement solutions. A partitioning-based approach integrates wirelength, temperature, TSV counts, and thermal effect into the min-cut objective [7]. As known for the 2-D placement problem, however, a partitioning-based approach is not as competitive as an analytical one. A multilevel analytical placement method is proposed in [3] for 3-D ICs to relax discrete layer assignment so that the movements of cells are continuous in the z-direction. Its basic idea is to use an interlayer density penalty function to remove cells between layers; however, the area occupied by TSVs is not considered during placement. A force-directed 3-D placement method was proposed recently [8]. A partitioning process is applied to assign cells to different layers, and then a force-directed quadratic algorithm is used to place cells and TSVs in a 3-D IC. Two design schemes are proposed to handle TSVs: 1) TSV-site: it places cells with regular fixed TSVs, and a TSV assignment stage then assigns these preplaced TSVs to cells, and 2) TSV co-placement: it simultaneously places TSVs and cells. The sizes of TSVs are considered; however, since no physical information is considered during partitioning, the placement solution quality is usually limited.

There is also an existing paper for the mixed-size placement that particularly considers large macros for 3-D IC designs [4]. Since this paper is focused more on the handling of big macros (instead of cells), it is beyond the scope of this paper.

B. Our Contributions

One of the common deficiencies in the previous works [3], [5], [7] is that the sizes of TSVs are not considered. As mentioned earlier, however, TSVs usually occupy significant areas and should be considered during 3-D IC placement. Traditionally, TSVs are inserted during the routing stage by searching whitespace in the whole 3-D IC, and thus the quality of a routing result strongly depends on the remaining whitespace after the placement stage. Fig. 2 illustrates the importance of considering whitespace reservation for TSVs during placement. If whitespace is not reserved for TSVs during placement, we observe that the available whitespace for TSV insertion is usually located along the chip periphery, especially for analytical placement; consequently, the resulting placement would incur longer wirelength since a TSV might be inserted in a whitespace far from its connected cells, as shown in Fig. 2(a). In contrast, with whitespace reservation for TSVs as illustrated in Fig. 2(b), a TSV can be inserted among cells to reduce the total wirelength.

Considering the physical locations of TSVs and their sizes and counts, we develop a new analytical cell placement algorithm for 3-D IC designs. The main contributions of this paper are summarized as follows.

1) A new 3-D cell placement algorithm that considers the sizes of TSVs and the physical positions for TSV insertion is proposed. This algorithm consists of three stages: a) 3-D global placement with density optimization and whitespace reservation for TSVs; b) TSV insertion and TSV-aware legalization; and c) layer-by-layer detailed placement.
2) A novel weighted-average (WA) wirelength model for analytical global placement is presented. Compared with the well-known log-sum-exp (LSE) wirelength model [9] that has dominated modern placement research for a decade, the proposed WA wirelength model gives the first model in the literature that can outperform the LSE one theoretically (with smaller estimation errors) and empirically. Also, a scheme is proposed to enhance the numerical stability of the WA wirelength model.
3) Instead of using cell areas to evaluate placement density, a new density cube is introduced to model the density of 3-D placement.
4) In addition to TSV count minimization, the density introduced by the sizes of TSVs are modeled in the 3-D analytical placement formulation. To the best of our knowledge, this is the first work that handles the sizes of TSVs during 3-D analytical placement with cell movement between layers.
5) A TSV insertion algorithm based on the overlapping whitespace area between neighboring layers is proposed to determine the location of every required TSV.
6) Since the physical positions of TSVs are determined during placement, 3-D routing can easily be accomplished with traditional 2-D routers. Compared with the state-of-the-art 3-D cell placement works [3], [8], our algorithm can achieve the best routed wirelength, TSV counts, and total silicon area, in shortest runtime.

The remainder of this paper is organized as follows. Section II formulates the 3-D placement problem. Section III presents the overall flow of the TSV-aware 3-D analytical placement. Sections IV and V detail the techniques used for the TSV-aware 3-D analytical placement. Section VI shows the experimental results. Finally, Section VII concludes this paper.

x_i, y_i	center coordinate of block v_i
z_i	layer of block v_i
$w_{b,k}, h_{b,k}$	width and height of bin b in layer k
$P_{b,k}$	area of pre-placed blocks in bin b of layer k
$D_{b,k}$	area of movable blocks in bin b of layer k
$T_{b,k}$	area of TSVs in bin b of layer k
$M_{b,k}$	the maximum free space in bin b of layer k
$t_{density}$	target placement density

Fig. 3. Notations used in this paper.

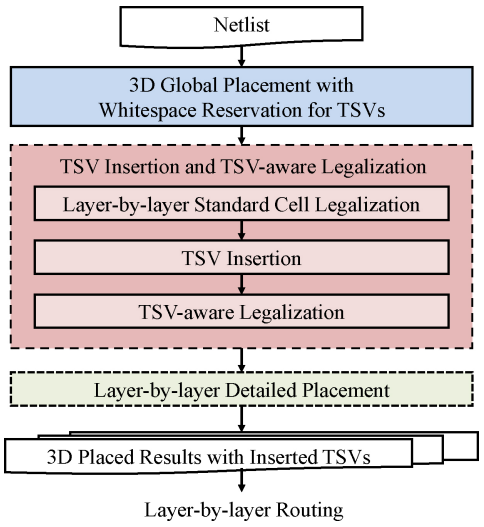


Fig. 4. Proposed TSV-aware 3-D analytical placement flow.

Algorithm: **3D Analytical Placement**
Input:
hypergraph H_0: circuit to be placed
n_{max}: the maximum block number in the coarsest level
W: the width of the placement region of the circuit
Output:
(x^*, y^*, z^*): 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 $QuadraticPlacement(H_{level})$;
06. initialize $\gamma \propto 0.05W$
07. **for** $currentLevel$ = $level$ **to** 0
08. initialize bin grid size $n_{bin} \propto \sqrt{n_x}$;
09. initialize $\lambda_0 = \frac{\sum |\partial \hat{W}(\mathbf{x},\mathbf{y},\mathbf{z})|}{\sum |\partial \hat{D}_b(\mathbf{x},\mathbf{y},\mathbf{z})|}$; $m = 0$;
10. **do**
11. call $WhitespaceReservationForTSV()$;
12. solve $\min(\hat{W}(\mathbf{x},\mathbf{y}) + \alpha \cdot \hat{Z}(\mathbf{z})) + \lambda_m \sum_b (\hat{D}_b(\mathbf{x},\mathbf{y},\mathbf{z}) - \hat{M}_b)^2$;
13. $m++$;
14. $\lambda_m = 2\lambda_{m-1}$;
15. **until** (spreading enough or no further improvement in wirelength)
16. call $LayerAssignment()$;
17. decluster and update block positions;
18. call $CellLegalization()$;
19. call $TSVInsertion()$;
20. call $TSVAwareLegalization()$;
21. call $LayerByLayerDetailedPlacement()$.

Fig. 5. Our 3-D analytical placement algorithm.

II. Problem Formulation

The 3-D placement problem 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 block v_i, and z_i be the device layer to which v_i belongs. Given a placement region and the number of device layers k, we intend to determine the optimal positions of movable blocks so that the total wirelength and the number of required TSVs are minimized while satisfying the nonoverlapping constraints among blocks and TSVs. Like traditional 2-D placement, the 3-D placement problem is usually solved in three stages [3]: 1) global placement; 2) legalization; and 3) detailed placement. Global placement evenly distributes the blocks and finds the best position and layer for each block to minimize the target cost (e.g., wirelength, TSV counts). Then, legalization removes all cell overlaps for each layer. Finally, detailed placement refines the 3-D placement solution. We summarize the notations used in this paper in Fig. 3.

III. TSV-Aware 3-D Analytical Placement

There are three stages in our TSV-aware 3-D placement method: 1) 3-D global placement with density optimization and whitespace reservation for TSVs; 2) TSV insertion and TSV-aware legalization; and 3) layer-by-layer detailed placement. Fig. 4 summarizes the flow of the proposed algorithm.

In 3-D analytical global placement, in addition to the 3-D placement for cells, we model the sizes of TSVs into density constraints such that after the 3-D global placement, the whitespace required by TSVs is reserved. In the TSV insertion and TSV-aware legalization stage, we first legalize cells in the circuit with minimum displacement, and then insert TSVs to their best positions such that the overlaps between cells and inserted TSVs are minimized. We fix the positions of TSVs after insertion, and legalization for cells is then applied to remove overlaps. In the layer-by-layer detailed placement stage, 2-D detailed placement techniques are applied to further improve the solution quality, such as cell matching for wirelength optimization and cell sliding for density optimization. TSV locations are fixed, and only cells are movable during detailed placement. Since cells have been assigned to different layers and TSVs have been inserted, the total wirelength of a net equals the summation of the wirelength of its sub-nets in each layer. (Note that the wirelength associated with TSVs has been considered with the TSV count.) The placement results with inserted TSVs can then be routed layer-by-layer by traditional 2-D routers.

Due to the increasing complexity of the placement problem, we use the multilevel framework [10], [11] for global placement to improve the scalability. In the multilevel framework, there are two main stages: 1) the coarsening stage and 2) the uncoarsening stage. Fig. 5 shows our proposed 3-D analytical placement algorithm. Lines 1–4 give the coarsening stage. The initial placement is generated by quadratic placement [12] in line 5. Lines 6 to 17 give the uncoarsening stage. Lines 18–

20 list the TSV insertion and TSV-aware legalization step and line 21 is for the layer-by-layer detailed placement.

During the coarsening stage, we adopt the best-choice clustering algorithm [13] to reduce the number of movable blocks. The key idea of the best-choice clustering algorithm is to identify the globally best pair of blocks or smaller clusters to cluster, and the clustering score is usually determined by the connectivity between each pair of blocks or smaller clusters, e.g., the total number of connected net between them. The maximum number of blocks in the coarsest level $n_{\max}$ is set to 6000 in our implementation. After clustering, the initial placement is obtained by quadratic placement [12]. Then, the placement problem is solved from the coarsest level to the finest level, where the placement for the current level provides the initial placement for the next level. In each level, the bin size is set according to the number of clusters, and the value of λ is initialized according to the strength of wirelength and density gradients. Then, the unconstrained minimization problem is solved by the conjugate gradient (CG) method. We detail each stage in the following sections.

IV. TSV-Aware 3-D Global Placement

In this section, we present the 3-D analytical global placement engine and discuss the whitespace reservation for TSVs.

A. 3-D Analytical Global Placement Engine

Since the analytical placement framework has been shown very effective for the 2-D placement problem, we shall extend this framework to solve the 3-D placement problem. After the placement region is divided into nonoverlapping uniform bin grids on each device layer, the 3-D analytical global placement problem can be formulated as a constrained optimization problem as follows:

$$\begin{aligned} \min \quad & W(\mathbf{x},\mathbf{y}) + \alpha \cdot Z(\mathbf{z}) \\ \text{s.t.} \quad & D_{b,k}(\mathbf{x},\mathbf{y},\mathbf{z}) + T_{b,k}(\mathbf{x},\mathbf{y},\mathbf{z}) \le M_{b,k}, \forall \text{bin b of layer k} \end{aligned} \tag{1}$$

where $W(\mathbf{x},\mathbf{y})$ is the wirelength function, $Z(\mathbf{z})$ is the number of used TSVs, α is a weighted number, $D_{b,k}(\mathbf{x},\mathbf{y},\mathbf{z})$, and $T_{b,k}(\mathbf{x},\mathbf{y},\mathbf{z})$ are the respective functions that are the total areas of movable blocks and TSVs in bin b of layer k, and $M_{b,k}$ is the maximum allowable area of movable blocks and TSVs in bin b of layer k. $M_{b,k}$ can be computed by $M_{b,k} = t_{\text{density}}(w_{b,k}h_{b,k} - P_{b,k})$, where t_{density} is a user-specified target density value for each bin, $w_{b,k}$ ($h_{b,k}$) is the width (height) of bin b of layer k, and $P_{b,k}$ is the area of preplaced block in bin b of layer k. Unlike traditional 2-D placers and previous 3-D placers [3], [5]–[7] that only consider the density $D_{b,k}$ of movable blocks, the sizes of TSVs are also considered in our formulation. Since the actual positions of TSVs are not determined, $T_{b,k}$ is a dynamic value during the placement process. $T_{b,k}$ will be explained in more detail in Section IV-B.

1) *Wirelength and TSV Models:* The wirelength $W(\mathbf{x},\mathbf{y})$ is defined as the total half-perimeter wirelength (HPWL)

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

Since the exact TSV positions are unknown during global placement, the number of TSVs is an estimation. There are two major types of TSVs: 1) via-first, and 2) via-last TSVs. While via-first TSVs interfere with device layer only, via-last TSVs interfere with both device and metal layers and should be aligned between neighboring device layers [8]. For both types of TSVs, the number of TSVs used for each net could be approximated by the number of layers it spans. Consequently, the number of TSVs $Z(\mathbf{z})$ is estimated through a similar way like wirelength [3]

$$Z(\mathbf{z}) = \sum_{\text{net } e} \max_{v_i,v_j \in e} |z_i - z_j|. \tag{3}$$

Since $W(\mathbf{x},\mathbf{y})$ in (2) is not smooth and nonconvex, it is hard to minimize it directly. As a result, several smooth wirelength approximation functions have been proposed, such as quadratic wirelength [12], CHKS wirelength [14], and LSE wirelength [9]. The LSE wirelength model

$$\begin{aligned} \gamma \textstyle\sum_{e \in E} (\ln \sum_{v_i \in e} \exp(x_i/\gamma) + \ln \sum_{v_i \in e} \exp(-x_i/\gamma) + \\ \ln \textstyle\sum_{v_i \in e} \exp(y_i/\gamma) + \ln \sum_{v_i \in e} \exp(-y_i/\gamma)) \end{aligned} \tag{4}$$

proposed in [9] often achieves the best result among recent 2-D academic placers [11]. When γ approaches zero, the LSE wirelength is close to the HPWL [9]. (Note that $Z(\mathbf{z})$ can be smoothed in a similar way.) Due to the computation precision, however, γ cannot be arbitrarily small to avoid arithmetic overflow, and thus an estimation error is inevitable.

Given a set of x coordinates for calculating the wirelength of a net e, $\mathbf{x}_e = \{x_i | v_i \in e\}$, let $\varepsilon_{LSE}(\mathbf{x}_e)$ be the estimation error of the LSE wirelength model with respect to the x coordinate. From [15], it is not difficult to derive the error bounds that $0 \le \varepsilon_{LSE}(\mathbf{x}_e) \le \gamma \ln n$, where n is the number of x coordinates. The error bounds for the y and z coordinates can be derived similarly.

2) *Weighted-Average Wirelength Model:* In this paper, we propose a novel WA wirelength model to approximate the respective maximum and minimum functions in (2) and (3) with smaller estimation errors than the LSE wirelength model. Given a set of x coordinates, $\mathbf{x}_e$, for calculating the wirelength of net e, the weighted average is given by

$$\bar{X}(\mathbf{x}_e) = \frac{\sum_{v_i \in e} x_i F(x_i)}{\sum_{v_i \in e} F(x_i)} \tag{5}$$

where $F(x_i)$ is the weighting function of x_i and is nonnegative. It is intuitive that $x_{\min} \le \bar{X}(\mathbf{x}_e) \le x_{\max}$, where $x_{\max}$ and $x_{\min}$ are the respective maximum and minimum values of $\mathbf{x}_e$.

To approximate the maximum value in $\mathbf{x}_e$, $F(x_i)$ should grow fast and can separate larger values from smaller ones. To achieve this goal, the exponential function is used

$$F(x_i) = \exp(x_i/\gamma) \tag{6}$$

where γ is the same as that in (4). Note that other functions with a similar property to the exponential function can also be used. The estimation function for the maximum value is then defined as

$$X_{\max}(\mathbf{x}_e) = \frac{\sum_{v_i \in e} x_i \exp(x_i/\gamma)}{\sum_{v_i \in e} \exp(x_i/\gamma)}. \tag{7}$$

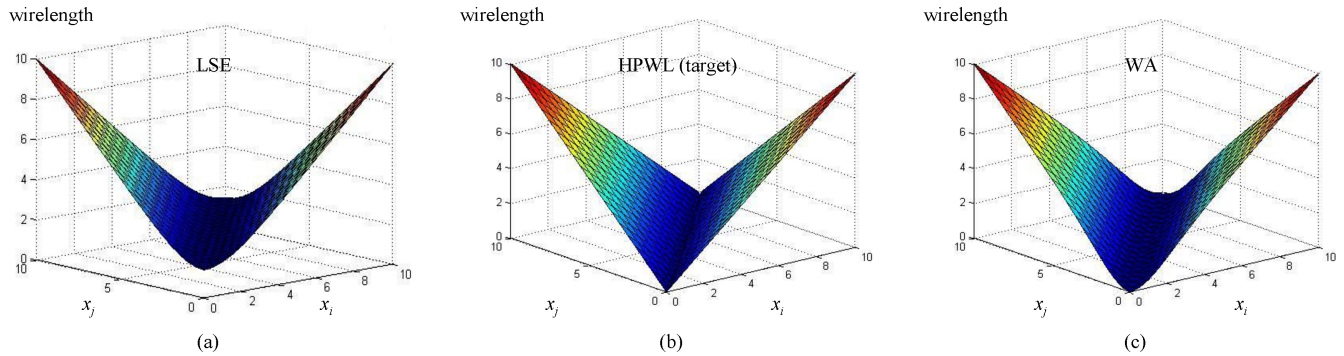


Fig. 6. MATLAB plots of wirelength models with two variables. (a) LSE wirelength. (b) HPWL. (c) WA wirelength.

The estimation function for the minimum value can be defined similarly. Therefore, the WA wirelength model is given by

$$\sum_{e\in E}\left(\frac{\sum_{v_i\in e} x_i \exp(x_i/\gamma)}{\sum_{v_i\in e}\exp(x_i/\gamma)} - \frac{\sum_{v_i\in e} x_i \exp(-x_i/\gamma)}{\sum_{v_i\in e}\exp(-x_i/\gamma)} + \frac{\sum_{v_i\in e} y_i \exp(y_i/\gamma)}{\sum_{v_i\in e}\exp(y_i/\gamma)} - \frac{\sum_{v_i\in e} y_i \exp(-y_i/\gamma)}{\sum_{v_i\in e}\exp(-y_i/\gamma)}\right). \tag{8}$$

The WA wirelength model converges to the HPWL in (2), as γ converges to 0. It is clear later that the WA wirelength model is continuously differentiable and smooth. Fig. 6 illustrates the LSE wirelength, the target HPWL, and our proposed WA wirelength with two variables, x_i and x_j, where horizontal axes represent the values of x_i and x_j, and the vertical axes represent the resulting wirelengths using different wirelength models.

Let $\varepsilon_{WA}(\mathbf{x}_e)$ be the estimation error of the WA wirelength model with respect to the x coordinate. We have the following estimation error bounds for this model.

Theorem 1: $0 \le \varepsilon_{WA}(\mathbf{x}_e) \le \dfrac{\gamma\Delta x}{1+\exp(\Delta x)/n}$, where $\Delta x = (x_{\max} - x_{\min})/\gamma$.

Proof 1: Let $x_{\max}(x_{\min})$ be the maximum (minimum) value of $\mathbf{x}_e$, and $X_{\max}(\mathbf{x}_e)$ $(X_{\min}(\mathbf{x}_e))$ be the estimation function for the maximum (minimum) value of $\mathbf{x}_e$. For the maximum function of the WA wirelength model, its estimation error is

$$\varepsilon^*_{WA}(\mathbf{x}_e) = x_{\max} - X_{\max}(\mathbf{x}_e) = \gamma\frac{\sum_{v_i\in e}\Delta x_i^* \exp(-\Delta x_i^*)}{\sum_{v_i\in e}\exp(-\Delta x_i^*)} \tag{9}$$

where $\Delta x_i^* = (x_{\max} - x_i)/\gamma$. By the definition of the weighted average, $X_{\max}(\mathbf{x}_e) \le x_{\max}$, and thus $\varepsilon^*_{WA}(\mathbf{x}_e) \ge 0$. By definition, $\mathbf{x}_e = \{x_1, x_2, ..., x_n\}$. Without loss of generality, let $x_1 \ge x_2 \ge \cdots \ge x_n$. After expanding (9), we have

$$\varepsilon^*_{WA}(\mathbf{x}_e) = \gamma\frac{\Delta x_{1,2}\exp(-\Delta x_{1,2}) + \Delta x_{1,3}\exp(-\Delta x_{1,3}) + \cdots + \Delta x_{1,n}\exp(-\Delta x_{1,n})}{1+\exp(-\Delta x_{1,2}) + \exp(-\Delta x_{1,3}) + \cdots + \exp(-\Delta x_{1,n})}. \tag{10}$$

To find the upper bound, we differentiate (10) with respect to variables $\Delta x_{1,2}, \Delta x_{1,3}, \ldots, \Delta x_{1,n}$ and make them 0s. For example, for any i

$$\begin{aligned}&\partial\varepsilon^*_{WA}(\mathbf{x}_e)/\partial\Delta x_{1,i} = 0 \leftrightarrow \\ &(1-\Delta x_{1,i})(1+\sum_{k=2}^{n}\exp(-\Delta x_{1,k}))+ \\ &\left(\sum_{k=2}^{n}\Delta x_{1,k}\exp(-\Delta x_{1,k})\right) = 0\end{aligned} \tag{11}$$

implying that $\Delta x_{1,2} = \Delta x_{1,3} = \cdots = \Delta x_{1,n}$, and $x_{\min} = x_2 = x_3 = \cdots = x_n$. Thus

$$\begin{aligned}0 \le \varepsilon^*_{WA}(\mathbf{x}_e) &\le \gamma\frac{(n-1)\Delta x\exp(-\Delta x)}{1+(n-1)\exp(-\Delta x)} = \frac{\gamma\Delta x}{1+\dfrac{\exp(\Delta x)}{n-1}} \\ &\le \frac{\gamma\Delta x}{1+\frac{\exp(\Delta x)}{n}}\end{aligned} \tag{12}$$

where $\Delta x = (x_{\max} - x_{\min})/\gamma$.

Similarly, we have the same bounds for the minimum function. Since $\varepsilon_{WA}(\mathbf{x}_e) = |(x_{\max} - x_{\min}) - (X_{\max}(\mathbf{x}_e) - X_{\min}(\mathbf{x}_e))| = (x_{\max} - X_{\max}(\mathbf{x}_e)) - (x_{\min} - X_{\min}(\mathbf{x}_e)) \le \dfrac{\gamma\Delta x}{1+\exp(\Delta x)/n}$, the theorem thus holds. (Note that $\varepsilon_{WA}(\mathbf{x}_e) \ge 0$.)

By mathematical induction, we have the following theorem:

Theorem 2: The estimation error upper bound of the WA wirelength model is smaller than that of the LSE wirelength model, i.e., $\frac{\gamma\Delta x}{1+\exp(\Delta x)/n} < \gamma\ln n, \forall n \ge 2$.

Fig. 7 shows the estimation errors for the LSE and our WA wirelength models. As shown in Fig. 7, our WA wirelength model has smaller estimation errors than those of the LSE

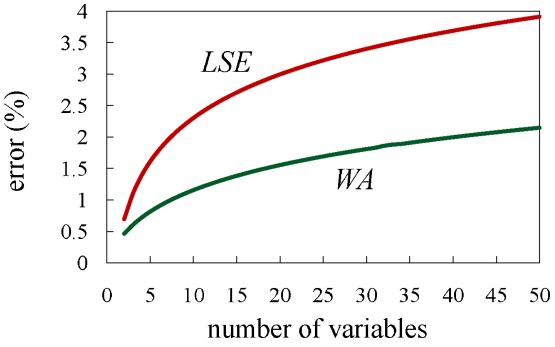


Fig. 7. Estimation error comparisons of the LSE wirelength model and the proposed WA wirelength model.

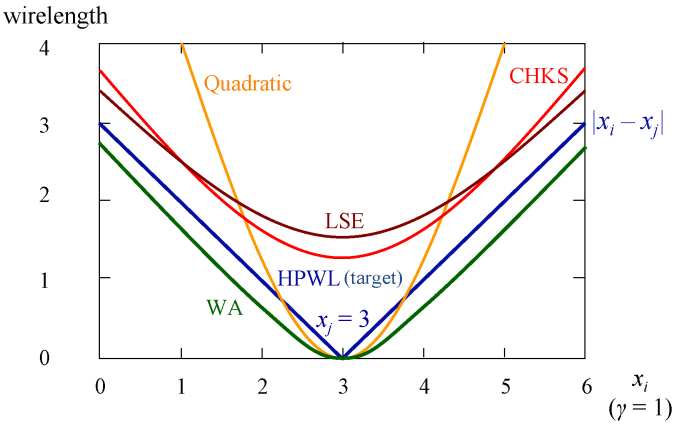


Fig. 8. Popular wirelength models with two variables.

wirelength model, especially when the number of variables grows. For example, the respective error bounds for the WA wirelength model for $n = 2, 3, 5, 10$ are 0.46, 0.60, 0.81, and 1.16γ, while those for the LSE one are 0.69, 1.10, 1.61, and 2.3γ, respectively.

Fig. 8 illustrates the comparisons among popular wirelength models with two variables, x_i and x_j, where the horizontal axis represents the value of x_i, and the vertical axis represents the resulting wirelengths using different wirelength models. Different from Fig. 6, x_j is assumed to be 3 for clearer comparisons between different models in Fig. 8. As shown in Fig. 8, the quadratic wirelength model cannot accurately approximate the HPWL when two terminals are far away. While the CHKS model approximates HPWL better than the LSE model when two terminals are closer, the LSE model gives better approximations when two terminals are farther. Most importantly, compared with the quadratic, CHKS, and LSE models, our WA wirelength model gives the best approximation to HPWL, not only for its nearness to the HPWL curve, but also for its trend of the function growth.

3) *Stable Weighted-Average Wirelength Model:* Due to the computation of exponential functions in (4) and (8), numerical instability might occur for the LSE and WA models when γ is too small. Kuwano and Takashima [16] used precomputed maximum and minimum values to reduce the computed values of the exponential functions and proposed a Stable-LSE model. The Stable-LSE model is theoretically effective; however, the extraction of the maximum and minimum values after the exponential and logarithm operations in LSE might be distorted during computations. Compared with the Stable-LSE model, the maximum and minimum values can easily be extracted and eliminated in our WA model. For example, the estimation function for the maximum value in the WA model can be stabilized by subtracting the maximum value as follows:

$$\begin{aligned}\frac{\sum_{v_i\in e} x_i \exp(\frac{x_i}{\gamma})}{\sum_{v_i\in e} \exp(\frac{x_i}{\gamma})} &= \frac{\exp(\frac{x_{\max}}{\gamma})\sum_{v_i\in e} x_i \exp(\frac{x_i - x_{\max}}{\gamma})}{\exp(\frac{x_{\max}}{\gamma})\sum_{v_i\in e} \exp(\frac{x_i - x_{\max}}{\gamma})} \\ &= \frac{\sum_{v_i\in e} x_i \exp(\frac{x_i - x_{\max}}{\gamma})}{\sum_{v_i\in e} \exp(\frac{x_i - x_{\max}}{\gamma})}.\end{aligned} \tag{13}$$

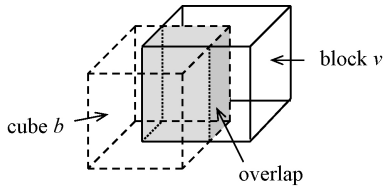


Fig. 9. Illustration of a density cube. The placement density is evaluated according to the overlaps among density cubes and blocks.

In this way, γ can be very small and thus enhance the robustness of the WA model.

4) *Density Model:* To evaluate the placement density, overlaps among bins and blocks are calculated. Unlike traditional 2-D placers [11], [17]–[20] and previous 3-D placers [3], [5]–[7] that usually use only horizontal and vertical overlaps to calculate the placement density for each layer, we introduce a density cube model to evaluate the density of 3-D placement. Adding the z dimension, rectangular bins and blocks become cubes and cuboid blocks, respectively. Note that the heights of cubes and cuboid blocks are directly proportional to the distance between two layers (unit length). The density of 3-D placement is then calculated by the overlaps among cubes and cuboid blocks in all the x, y, and z directions. Fig. 9 shows an example of the density cube model. The density of a cube b of layer k can be defined as

$$D_{b,k}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{v\in V}(P_x(b, v, k)P_y(b, v, k)P_z(b, v, k)) \tag{14}$$

where $P_x(b, v, k)$, $P_y(b, v, k)$, and $P_z(b, v, k)$ denote the overlaps between block v and cube b of layer k along the x, y, and z directions, respectively. In such a way, blocks can be distributed evenly among layers under the density constraints. The bell-shaped function [19] can be extended to transform the overlap function $D_{b,k}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ into a smooth and differentiable function for our analytical placement.

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

$$\min \quad W(\mathbf{x}, \mathbf{y}) + \alpha Z(\mathbf{z}) + \lambda \sum_{b,k}(\hat{D}_{b,k}(\mathbf{x}, \mathbf{y}, \mathbf{z}) + T_{b,k} - M_{b,k})^2 \tag{15}$$

with increasing λs. The solution of the previous problem is used as the initial solution for the next one. We solve the unconstrained problem in (15) by the CG method.

B. Whitespace Reservation for TSVs

To estimate the spaces occupied by TSVs, in addition to cell density $D_{b,k}$, the density of TSVs $T_{b,k}$ is also added to the density constraints for global placement. Since the actual positions of TSVs are not determined during global placement, $T_{b,k}$ is a dynamic value. A reasonable assumption is that the communication between neighboring layers of a net is through one TSV. The *net-box* is defined as the range spanned by a net. Just like traditional 2-D routing, for a net, placing the corresponding vias inside its net-box leads to the fewest routing detours. See Fig. 10 for an illustration. Given a net

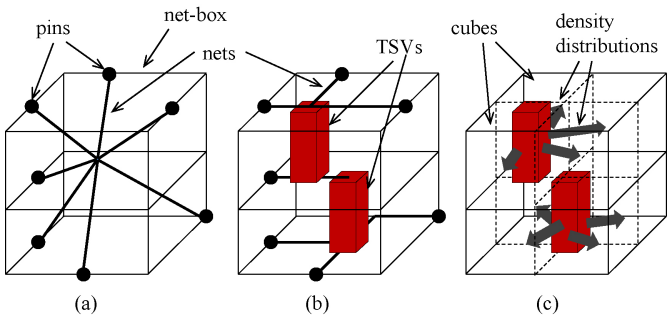


Fig. 10. Illustration of whitespace reservation during 3-D global placement. (a) Net and its corresponding net-box. (b) Two TSVs are needed for this net. (c) Required spaces for TSVs are distributed into cubes inside the net-box evenly.

and its connected pins, we can derive its net-box. Since the net shown in Fig. 10 spans three layers, two TSVs are needed for this net. We distribute required spaces for TSVs into density cubes inside the net-box evenly, such that there are enough spaces for TSV insertion inside this net-box, as shown in Fig. 10(c).

After 3-D analytical global placement, the amounts of whitespace needed for TSV insertion are reserved as much as possible.

C. Layer Assignment

At the end of the 3-D global placement, cells are evenly distributed in the 3-D placement region. To finalize the placement layers of cells, from the lowest layer to the highest layer, cells are assigned to the nearest layer, without exceeding the cell capacity which equals die area times cell utilization rate.

V. TSV Insertion and TSV-Aware Legalization

In this step, we attempt to insert TSVs among legalized cells without overlaps. Since the remaining whitespace after 3-D global placement may not be enough for inserting TSVs, we use a three-step scheme to insert required TSVs and legalize both cells and TSVs such that there are no cell-to-cell, TSV-to-TSV, cell-to-TSV overlaps. First, we legalize cells in each layer with minimum displacement without considering TSVs. Second, a greedy TSV insertion method is applied to insert required TSVs while minimizing cell-to-TSV overlaps. Finally, post-legalization is applied to remove cell-to-TSV and cell-to-cell overlaps with TSVs being fixed. We detail each step as follows.

A. Layer-by-layer Standard Cell Legalization

To perform the layer-by-layer standard-cell legalization, traditional 2-D legalization techniques, such as [21], [22], can be applied to each layer. Unlike 2-D legalization, however, connections among different layers need to be considered during the legalization for 3-D ICs.

B. TSV Insertion

Fig. 11 shows the overall flow of our TSV insertion algorithm. Given a legalized placement, we try to insert TSVs to the positions with minimum overlaps with legalized cells

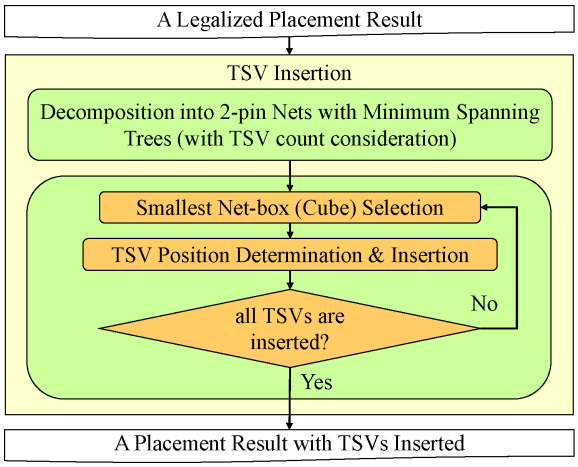


Fig. 11. Overall flow of the TSV insertion algorithm.

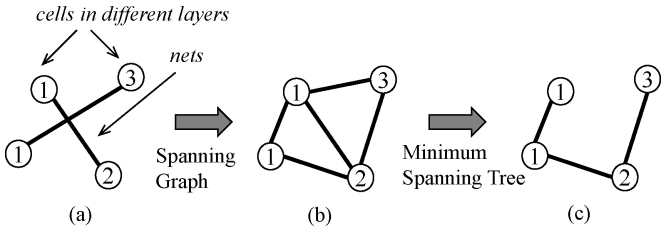


Fig. 12. MST generation. Here, the number in each vertex represents its corresponding layer. (a) Connection of one net. (b) Spanning graph construction for the net. (c) MST generation for the net.

such that, after TSV insertion, the total movement of cells for overlap removal is minimized.

Initially, each net is decomposed into 2-pin nets by a minimum spanning tree (MST) algorithm, and we insert TSVs for a net at one time. The TSV insertion proceeds in a nondecreasing order of the net-box sizes of 2-pin nets. Since a bigger net-box typically has a larger whitespace area in the net-box for TSV insertion, the TSV insertion process starts from the 2-pin net with the smallest net-box. In the following, we detail the MST generation and TSV position determination.

1) *Minimum Spanning Tree (MST) Generation:* Fig. 12 shows an example of the MST generation for a net. In Fig. 12(a), a multipin net is chosen, where each vertex represents a cell connected by this net and the number in each vertex represents its corresponding layer. We adopt the spanning graph approach in [23] which is efficient and effective for constructing spanning trees [see Fig. 12(a) and(b)]. Note that all of the cells are projected to a single layer such that the geometric relation in the spanning graph is captured by the planar distance between cells. Different from traditional spanning tree construction, TSV counts are also considered as costs for the construction. The cost of an edge e is defined as $\beta \cdot L(e) + \delta \cdot Z(e)$, where $L(e)$ is the planar wirelength of e, $Z(e)$ is the number of TSVs required for e, and β and δ are user-specified parameters. Kruskal's algorithm is then applied to construct an MST. It should be noted that, since we use only one TSV on each layer for each net, a large δ is adopted such that edges connecting between vertices in the same layer can be selected first.

TABLE I
RESULTING HPWLS, TSV NUMBERS, AND CPU TIME OF OUR PLACER AND THE 3-D PLACER [3] ON THE IBM-PLACE BENCHMARKS*

			Cong and Luo [3]			Our placer		
Circuit	No. of Cells	No. of Nets	WL ($\times 10^7$)	No. of TSV ($\times 10^3$)	Time* (min)	WL ($\times 10^7$)	No. of TSV ($\times 10^3$)	Time (min)
ibm01	12k	12k	0.37	0.87	7.19	0.33	0.57	0.40
ibm03	22k	22k	0.84	2.92	12.31	0.76	2.76	0.90
ibm04	27k	26k	1.11	3.36	24.21	0.99	2.53	1.20
ibm06	32k	33k	1.45	3.40	27.07	1.23	3.97	1.40
ibm07	45k	44k	2.27	4.46	36.00	1.87	4.95	2.60
ibm08	51k	48k	2.36	4.43	30.81	2.02	4.62	2.40
ibm09	52k	50k	2.08	3.37	30.14	1.85	3.27	2.30
ibm13	82k	84k	4.14	4.37	45.29	3.34	3.83	3.90
ibm15	158k	161k	8.74	27.53	114.04	7.61	15.56	9.90
ibm18	210k	201k	12.88	38.35	156.42	11.34	12.21	21.30
Average			1.00	1.00	1.00*	0.87	0.84	0.08

*The experiments in [3] were conducted on a Linux machine with AMD Opteron 1.8 GHz CPUs and 8 GB memory.

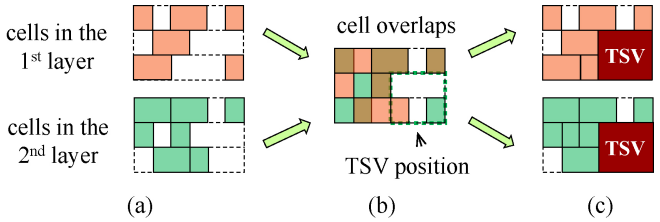


Fig. 13. Illustration of TSV position determination. (a) Placement result of two neighboring layers for TSV insertion. (b) Overlapping view of the placement result between these two layers. (c) Required TSVs are inserted according to the overlapping whitespace area.

2) *TSV Position Determination:* After each net has been decomposed into 2-pin nets, we start to insert required TSVs into the placement region. We sort the 2-pin nets in a nondecreasing order of their net-box sizes, and at each time, the 2-pin net with the smallest net-box is chosen for TSV insertion. In this paper, we consider the alignment constraint for TSVs. As mentioned in Section IV-A1, there are two major types of TSVs. Via-last TSVs that interfere with both device and metal layers should be aligned between neighboring device layers, and thus are more restricted. In contrast, via-first TSVs interfere with device layer only, and thus can be inserted layer-by-layer independently; with the alignment consideration of via-first TSVs, however, interconnections between TSVs of neighboring layers can be minimized.

For each net, we first evaluate the whitespace on each neighboring layer spanned by its net-box. Then, we calculate the overlapping whitespace area among these layers enclosed by the net-box. Fig. 13 illustrates an example. The regions enclosed by the net-box are further divided into smaller bins such that at most one TSV can be inserted into a bin. Note that, if the minimum spacing constraints for TSVs are considered, the bin size should be slightly larger than the TSV size, such that TSVs can be inserted into bins without violating the spacing constraints. After calculating the overlapping whitespace area [see Fig. 13(a) and (b)], the TSV positions are decided by searching a bin in the enclosed region such that the overlaps between cells and inserted TSVs are minimized, and there is no overlap between any two TSVs, as shown in Fig. 13(c). If there is not enough whitespace in the net-box, the searched region is doubled, and the search process continues.

C. TSV-Aware Legalization

After TSVs are inserted, postlegalization is applied to remove overlaps between cells and TSVs. The legalization techniques presented in Section V-A are applied again; at this time, however, TSVs are considered as fixed blockages when performing legalization.

VI. EXPERIMENTAL RESULTS

We conducted four experiments to evaluate our algorithm. All experiments were performed on the same PC workstation with eight Intel Xeon 2.5 GHz CPUs and 26 GB memory. We implemented our algorithm in the C++ programming language and integrated our code into NTUplace3 [11], which is a leading academic 2-D placer. We also modified the legalizer and detailed placer of NTUplace3 to support the layer-by-layer legalization and detailed placement in our proposed 3-D placement flow. The weight of TSV counts in (15), α, was set to 10 as in [3], β and δ for minimum spanning construction were set to 0.4 and 0.6, respectively, and the γ value for each circuit was set to $0.05 \times W$, where W is the width of the placement region of the circuit.

In the first experiment, we examined the quality of our 3-D analytical placement engine by comparing with the state-of-the-art 3-D analytical placer [3]. (Note that the *mixed-size* placer [4] is focused more on the handling of big macros and applies the 3-D placer [3] for standard-cell placement; for fair comparison, we thus should compare with [3] directly.) The second experiment compared with a recent force-directed 3-D placer with TSV area consideration [8]. In the third experiment, we examined the effectiveness of our whitespace reservation during global placement. The fourth experiment compared our WA wirelength model with the LSE one. We report the results in the following subsections.

TABLE II

RESULTING ROUTED WIRELENGTH (WL_R), TSV NUMBERS (#TSV), TOTAL SILICON AREA (AREA), CPU TIME FOR PLACEMENT ($TIME_P$) AND ROUTING ($TIME_R$), AND TOTAL CPU TIME ($TIME_T$) OF OUR PLACER AND THE FORCE-DIRECTED 3-D PLACER [8] ON THE IWLS AND INDUSTRY BENCHMARKS

			Kim *et al.* [8]				Our placer			
Circuit	No. of Cells	No. of Nets	WL_R ($\times 10^5 \mu$m)	No. of TSV ($\times 10^3$)	Area ($\times 10^4 \mu m^2$)	$Time_P$* (min)	WL_R ($\times 10^5 \mu$m)	No. of TSV ($\times 10^3$)	Area ($\times 10^4 um^2$)	$Time_P$ (min)
Ind1	11k	12k	4.00	1.70	6.97	1.55	3.35	1.63	5.87	0.31
Ind2	15k	15k	2.84	1.30	5.86	0.88	2.59	1.04	5.20	0.58
Ind3	16k	16k	3.01	0.80	6.97	1.35	2.39	0.66	5.25	0.57
Ind4	20k	20k	3.88	1.02	8.07	1.68	3.87	1.05	6.88	0.64
Ind5	30k	30k	5.83	2.79	14.75	3.13	5.52	2.58	12.09	0.75
ethernet	77k	77k	14.01	3.87	34.11	21.45	13.51	3.11	29.22	4.43
RISC	88k	89k	20.02	4.44	38.69	12.12	17.87	2.12	30.17	6.90
b18	104k	104k	26.83	10.40	49.56	18.90	18.27	1.43	31.80	5.53
des_perf	109k	109k	19.12	3.86	38.69	15.83	22.91	5.53	35.75	7.96
b19	169k	169k	39.46	8.50	71.23	36.22	29.80	4.16	58.66	13.27
Average			1.00	1.00	1.00	1.00*	0.90	0.79	0.82	0.38

*The experiments in [8] were conducted on a Linux machine with eight Intel Xeon 2.5 GHz CPUs and 16 GB memory.

TABLE III

RESULTING ROUTED WIRELENGTH (WL_R), TSV NUMBERS (#TSV), TOTAL SILICON AREA (AREA), CPU TIME FOR PLACEMENT ($TIME_P$) AND ROUTING ($TIME_R$), AND TOTAL CPU TIME ($TIME_T$) FOR OUR PLACER WITHOUT AND WITH WHITESPACE RESERVATION ON THE IWLS AND INDUSTRY BENCHMARKS

	Our method without whitespace reservation						Our method with whitespace reservation					
Circuit	WL_R ($\times 10^5 \mu$m)	No. of TSV ($\times 10^3$)	Area (μm^2)	$Time_P$ (min)	$Time_R$ (min)	$Time_T$ (min)	WL_R ($\times 10^5 \mu$m)	No. of TSV ($\times 10^3$)	Area (μ^2)	$Time_P$ (min)	$Time_R$ (min)	$Time_T$ (min)
Ind1	3.23	1.70	6.38	0.24	2.93	3.17	3.35	1.63	5.87	0.31	2.90	3.21
Ind2	2.61	1.10	5.20	0.54	2.71	3.25	2.59	1.04	5.20	0.58	2.69	3.27
Ind3	2.56	0.65	5.38	0.54	2.87	3.41	2.39	0.66	5.25	0.57	2.90	3.46
Ind4	3.66	1.07	6.88	0.69	3.63	4.32	3.87	1.05	6.88	0.64	3.67	4.31
Ind5	5.81	2.60	12.32	0.75	5.48	6.23	5.52	2.58	12.09	0.75	5.38	6.13
ethernet	14.99	3.23	29.22	4.29	14.54	18.83	13.51	3.11	29.22	4.43	13.95	18.38
RISC	17.64	2.13	30.61	5.96	18.38	24.34	17.87	2.12	30.17	6.90	18.15	25.04
b18	18.46	1.46	32.42	6.33	18.54	24.87	18.27	1.43	31.80	5.53	17.94	23.46
des_perf	23.67	5.58	35.84	8.48	14.54	23.01	22.91	5.53	35.75	7.96	17.67	25.63
b19	34.33	4.19	59.56	15.66	33.06	48.72	29.80	4.16	58.66	13.27	25.51	38.78
Average	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.98	0.98	1.02	0.99	0.99

A. 3-D Analytical Placement Comparisons

We compared with the 3-D analytical placer [3] based on the IBM-PLACE benchmarks [24] used in [3]. As in [3], a four-layer implementation of 3-D IC was assumed, and the floorplan size was scaled by dividing the original area by 4, and then each layer was enlarged to obtain 10% whitespace. Note that, since the work [3] does not consider TSV area, TSV insertion is not applied in our placer here for fair comparison.

Table I summarizes the experimental results, where total wirelength (WL), numbers of TSVs, and runtime are compared. Columns 4–6 give the experimental results reported in [3], and Columns 7–9 list our results. Compared to [3], our algorithm can effectively reduce the wirelength and TSV counts by 13% and 16%, respectively. Although the work [3] was implemented on a Linux machine with AMD Opteron 1.8 GHz and 8 GB memory, the results also reveal that our algorithm is more efficient.

The work [3] uses an additional interlayer overlap function and unconnected filler cells for density control, which enlarges the placement problem size and complexity, while our algorithm directly optimizes the wirelength and TSV counts under density constraints by using the density cube model to spread blocks in the whole 3-D chip. The experimental results justify the effectiveness and efficiency of our placement algorithm.

B. TSV-Aware Placement Comparison

Based on the IWLS [25] and industry benchmarks used in [8], we also compared with the state-of-the-art force-directed 3-D placer [8] which considers TSV sizes during placement. As in [8], 45 nm technology and via-first TSVs were used, TSV cell size was set to 2.47 μm × 2.47 μm, 4-layer implementation of 3-D IC was assumed, and Cadence SoC Encounter [26] was used to route each layer after 3-D placement. For each layer, landing pads were inserted as the top mental wire to connect TSVs of the layer above, such that interconnections among layers can be complete. For fair comparison, TSV pitches are not considered as in [8]; however, we note that TSV pitches can easily be modelled by changing the sizes of TSVs, which will not affect our algorithm.

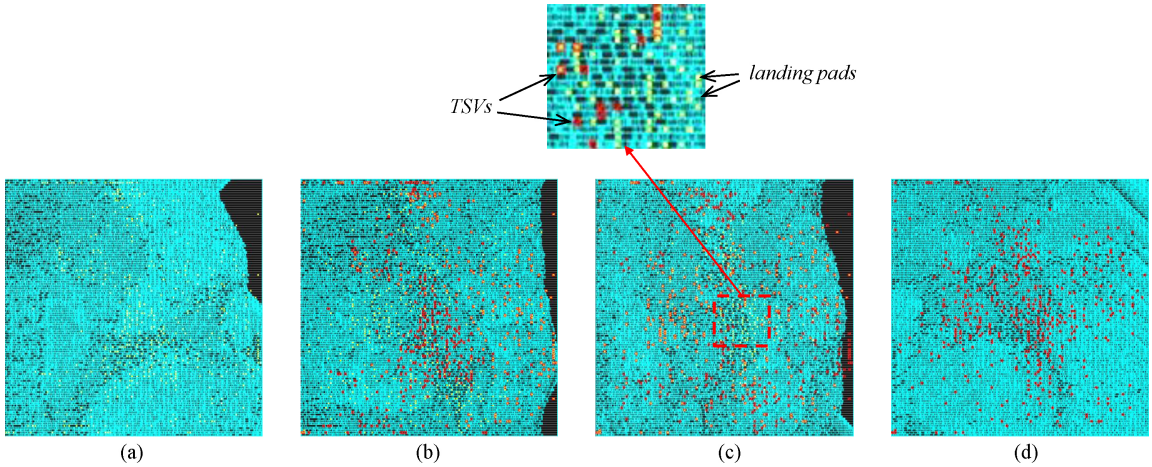


Fig. 14. Cadence SoC Encounter snapshot of each layer of the circuit b18, generated by our placer. Compared with [8], our placer achieves 32% shorter final routed wirelength, 86% fewer TSV counts, and 36% smaller total silicon area.

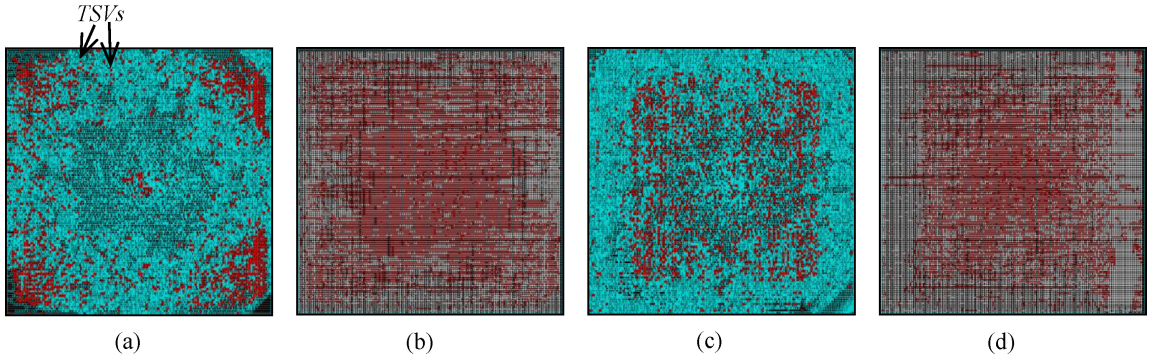


Fig. 15. Cadence SoC Encounter snapshots for the TSV insertion results of the second layer of the circuit des_perf, generated from our placer. (a) Placement result without whitespace reservation. (b) Routing congestion map without whitespace reservation, where red lines represent routing overflow regions while white lines represent nonoverflow ones. (c) Placement result with whitespace reservation. (d) Routing congestion map with whitespace reservation. With whitespace reservation during placement, TSVs are often inserted near the center of the placement region, leading to shorter routed wirelength and smaller routing congestion.

TABLE IV
RESULTING ROUTED WIRELENGTH (WL_R), TSV NUMBERS (#TSV), TOTAL SILICON AREAS (AREA), CPU TIME FOR PLACEMENT ($TIME_P$) AND ROUTING ($TIME_R$), AND TOTAL CPU TIME ($TIME_T$) FOR OUR PLACER WITHOUT WHITESPACE RESERVATION BY USING THE LSE MODEL ON THE IWLS AND INDUSTRY BENCHMARKS

	LSE wirelength model					
Circuit	WL_R ($\times 10^5$um)	No. of TSV ($\times 10^3$)	Area ($\times 10^4$um^2)	$Time_P$ (min)	$Time_R$ (min)	$Time_T$ (min)
Ind1	3.33	1.89	6.38	0.22	2.77	2.98
Ind2	2.54	1.02	5.20	0.57	2.60	3.17
Ind3	2.50	0.65	5.38	0.58	2.65	3.23
Ind4	3.65	1.13	6.88	0.70	3.28	3.98
Ind5	6.02	2.62	12.32	0.78	5.35	6.13
ethernet	15.75	3.18	29.22	4.90	14.28	19.18
RISC	18.07	2.09	30.61	6.27	18.82	25.08
b18	18.64	1.47	31.80	5.83	18.48	24.32
des_perf	23.43	5.52	35.84	6.85	19.60	26.45
b19	35.72	4.16	59.56	12.37	32.40	44.77
Average*	1.01	1.01	1.00	0.98	1.00	0.99

*The results are normalized by those of the WA model in Table III.

Table II summarizes the experimental results, where routed wirelength, numbers of TSVs, total silicon area, runtime for placement are compared. Columns 4–7 give the experimental results reported in [8], and Columns 8–11 list our results. As shown in the table, our algorithm can achieve on average 10% shorter wirelength, 21% fewer TSV counts, 18% smaller total silicon area, and significantly shorter runtime, compared with the force-directed quadratic placement algorithm [8]. Especially for larger circuits, our algorithm can achieve even better results than [8], implying that our algorithm has better scalability. (Note that the work [8] conducted its experiments on a Linux machine with eight Intel Xeon 2.5 GHz CPUs and 16 GB memory, similar to our environment except that our machine has 26 GB memory.) Fig. 14 shows the placement layouts of the b18 circuit generated by our placer. For b18, our placer achieves 32% shorter final routed wirelength, 86% fewer TSV counts, and 36% smaller total silicon area. The quality differences might lie in the fact that the work [8] applies partitioning for layer assignment before placement. Since there is no physical information during the partitioning, its placement quality is usually limited.

TABLE V
RESULTING HPWLS AFTER GLOBAL PLACEMENT (GP HPWL) AND DETAILED PLACEMENT (DP HPWL), AND CPU TIME BY USING THE LSE AND WA MODELS ON THE ISPD'06 (2-D) PLACEMENT CONTEST BENCHMARKS

			LSE wirelength model			WA wirelength model		
Circuit	No. of Cells	No. of Nets	GP HPWL ($\times 10^7$)	DP HPWL ($\times 10^7$)	CPU (min)	GP HPWL ($\times 10^7$)	DP HPWL ($\times 10^7$)	CPU (min)
adaptec5	842k	867k	36.55	36.05	65.85	36.91	35.20	63.78
newblue1	330k	338k	5.84	5.90	14.37	5.77	5.87	22.80
newblue2	436k	465k	18.84	19.13	34.42	19.22	19.27	36.43
newblue3	482k	552k	28.80	27.79	25.98	27.34	26.91	24.00
newblue4	642k	637k	26.33	25.40	54.97	24.87	24.06	67.20
newblue5	1228k	1284k	42.28	41.78	133.37	42.12	41.94	102.13
newblue6	1248k	1288k	55.49	53.71	89.70	49.29	48.34	151.48
newblue7	2481k	2636k	114.83	109.86	191.42	108.20	104.89	260.92
Average			1.00	1.00	1.00	0.98	0.97	1.20

Density targets are set to 1.0 for all circuits.

C. Whitespace Reservation Comparison

In this experiment, we examined the effectiveness of our whitespace reservation method during global placement. We compared our method with and without whitespace reservation. Table III lists the results. As mentioned before, the sizes of TSVs are significantly large, and TSVs are usually placed at the whitespace between macro blocks or standard cells and might affect the routing resource. If the whitespace for TSVs is reserved earlier, it can facilitate later routing. Our whitespace reservation can effectively reserve whitespace for TSVs and thus reduce the final routed wirelength. Besides, our whitespace reservation considers the density of TSVs during global placement, and it can also reduce the total number of required TSVs by 2% and the total silicon area by 2%. As shown in Table III, our method with whitespace reservation requires shorter runtime for routing and achieves 3% shorter final routed wirelength. Although performing the whitespace reservation for TSVs during global placement inevitably incurs overheads on the runtime for global placement, it reserves appropriate whitespace for TSV insertion and thus significantly facilitates the routing process.

Fig. 15 shows the TSV insertion results and the routing congestion maps of the second layer of circuit des_perf without and with whitespace reservation. The results reveal that TSVs are often inserted at whitespace far from the center of the placement region if no whitespace is reserved during global placement [see Fig. 15(a)], while many more TSVs are inserted close to the center of the placement region with the whitespace reservation [see Fig. 15(c)]. As shown in Fig. 15(b) and (d), routing congestion can substantially be reduced with whitespace reservation. The results show the effectiveness of our whitespace reservation.

D. Wirelength Model Comparison

In Section IV-A2, we showed that the WA wirelength model has smaller estimation error upper bound than the LSE one. In this experiment, we examined the empirical results for the two models, based on the IWLS and industry benchmarks. To reduce the control factors for fair comparison, the whitespace reservation algorithm was not applied, the original WA model was used to compare with the LSE model, and the same γ values were used for both models. Table IV lists the placement results of the LSE model. As shown in Table IV, our WA model can achieve on average 1% shorter final routed wirelength and 1% fewer TSV counts than the LSE model.

TABLE VI
RESULTING NORMALIZED HPWLS USING THE WA AND STABLE-WA MODELS WITH DIFFERENT γ VALUES ON THE ISPD'06 PLACEMENT CONTEST BENCHMARKS [27]

Circuit	WA		Stable-WA	
Name	$\gamma = 50$	$\gamma = 5$	$\gamma = 50$	$\gamma = 5$
adaptec5	1.00	17.17	0.95	0.93
newblue1	1.00	13.20	1.01	1.00
newblue2	1.00	13.38	1.00	0.98
newblue3	1.00	N/A	1.00	1.00
newblue4	1.00	N/A	1.02	1.00
newblue5	1.00	N/A	1.00	0.99
newblue6	1.00	20.11	0.99	0.98
newblue7	1.00	N/A	1.00	0.95
Average	1.00	N/A	1.00	0.98

Density targets are set to 1.0 for all circuits.

Note that the comparison listed in Table IV is based on the 3-D placement benchmarks, which consider TSV distribution. To focus more on the wirelength model comparison, we also compared with the LSE model based on the ISPD'06 placement contest benchmarks [27] with the numbers of cells ranging from 330k to 2481k. We integrated both the LSE and WA models into NTUplace3 [11]. Table V shows the experimental results. As shown in the table, our WA model can achieve on average 2% and 3% shorter total wirelength than the LSE model after global placement and detailed placement, respectively. Figs. 16 and 17 show the comparisons of resulting HPWLs and CPU time of the circuits newblue1 and newblue7 with different γ values, respectively. As shown in Figs. 16 and 17, our WA model can effectively achieve better placement results than the LSE model with small runtime overheads. It should be noted that the LSE model has been shown to be the best wirelength models currently available, and it has dominated the placement research for more than one decade; the 3% improvement is particularly significant, as it reveals that our WA model is the first model that can beat LSE theoretically and empirically.

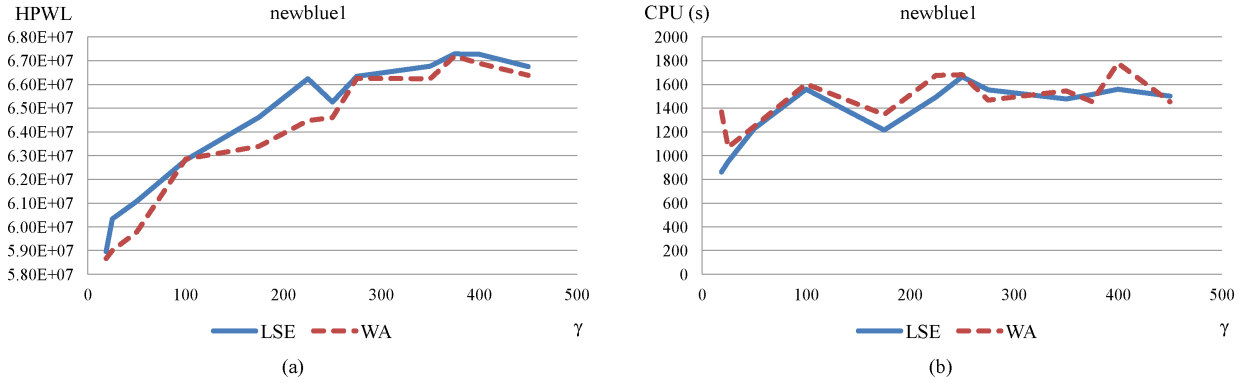


Fig. 16. Comparisons of resulting (a) HPWLs and (b) CPU time of the circuit newblue1 with different γ values.

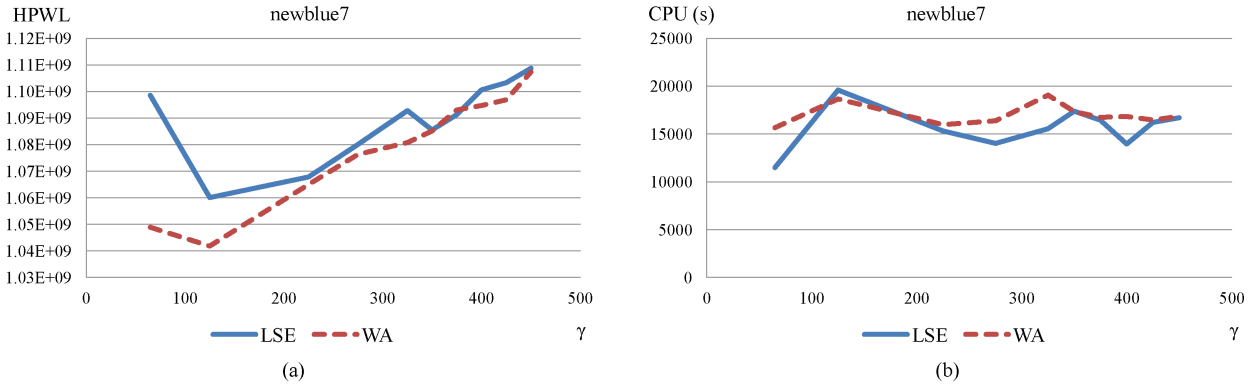


Fig. 17. Comparisons of resulting (a) HPWLs and (b) CPU time of the circuit newblue7 with different γ values.

We also examined the effectiveness of the Stable-WA model in Table VI. We used two different γ values, 50 and 5, for comparisons. As shown in Table VI, while the original WA model might lead to unacceptable placement results with a very small γ value, the Stable-WA model can achieve a better result even with a very small γ value, which shows the better robustness of the Stable-WA model.

VII. Conclusion

We presented a new TSV-aware placement algorithm for 3-D IC designs. The global placement was based on a novel WA wirelength model; to the best of our knowledge, it is the first model in the literature that can outperform the well-known LSE wirelength model theoretically and empirically. Unlike the previous works that only minimize the TSV count during placement, our algorithm additionally considers the whitespace reservation for TSVs and the physical positions for TSV insertion. Since the whitespace for TSVs is reserved and the physical positions of TSVs are determined during placement, 3-D routing can easily be accomplished by traditional 2-D routers. Experimental results showed that our algorithm can achieve the best routed wirelength, TSV counts, and total silicon area among all published works, with the shortest runtime.

As a relatively new technology, 3-D ICs open up many future research directions. For modern IC designs, there may be hundreds of large macros with millions of standard cells on a single chip, an effective 3-D placement technique for such mixed-size designs is desirable. Furthermore, the heat dissipation problem has been one of the most critical problems in 3-D ICs, and leakage power can be exponentially increased as temperature increases. Consequently, the power-delivery and thermal issues should be considered during 3-D placement. Finally, TSV-induced issues, such as thermal delivery or stress for manufacturability, should also be addressed during 3-D placement for reliable and manufacturable 3-D IC designs.

References

[1] M.-K. Hsu, Y.-W. Chang, and V. Balabanov, "TSV-aware analytical placement for 3-D IC designs," in *Proc. ACM/IEEE Design Autom. Conf.*, 2011, pp. 664–669.

[2] H.-S. Ye, M. C. Chi, and S.-H. Huang, "A design partitioning algorithm for 3-D integrated circuits," in *Proc. IEEE Int. Symp. Comput. Commun. Control Autom.*, May 2010, pp. 229–232.

[3] J. Cong and G. Luo, "A multilevel analytical placement for 3-D ICs," in *Proc. IEEE/ACM Asia South Pacific Design Autom. Conf.*, Jan. 2009, pp. 361–366.

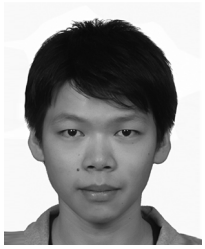
[4] J. Cong and G. Luo, "An analytical placer for mixed-size 3-D placement," in *Proc. ACM Int. Symp. Phys. Design*, Mar. 2010, pp. 61–66.

[5] J. Cong, G. Luo, J. Wei, and Y. Zhang, "Thermal-aware 3-D IC placement via transformation," in *Proc. IEEE/ACM Asia South Pacific Design Autom. Conf.*, Jan. 2007, pp. 780–785.

[6] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3-D ICs using a force directed approach," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2003, pp. 86–89.

[7] B. Goplen and S. Sapatnekar, "Placement of 3-D ICs with thermal and interlayer via considerations," in *Proc. ACM/IEEE Design Autom. Conf.*, 2007, pp. 626–631.

[8] D. H. Kim, K. Athikulwongse, and S. K. Lim, "A study of through-silicon-via impact on the 3-D stacked IC layout," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2009, pp. 674–680.
[9] 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, 2001.
[10] T. F. Chan, J. Cong, T. Kong, and J. R. Shinnerl, "Multilevel optimization for large-scale circuit placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2000, pp. 171–176.
[11] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 27, no. 7, pp. 1228–1240, Jul. 2008.
[12] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 10, no. 3, pp. 356–365, Mar. 1991.
[13] C. Alpert, A. Kahng, G.-J. Nam, S. Reda, and P. G. Villarrubia, "A semi-persistent clustering technique for VLSI circuit placement," in *Proc. ACM Int. Symp. Physical Design*, 2005, pp. 200–207.
[14] C. Li and C.-K. Koh, "Recursive function smoothing of half-perimeter wirelength for analytical placement," in *Proc. IEEE/ACM Int. Symp. Quality Electron. Design*, Mar. 2007, pp. 829–834.
[15] L.-T. Wang, Y.-W. Chang, and K.-T. Cheng, *Electronic Design Automation: Synthesis, Verification, and Test (Systems on Silicon)*. San Mateo, CA: Morgan Kaufmann, 2009.
[16] M. Kuwano and Y. Takashima, "Stable-LSE based analytical placement with overlap removable length," in *Proc. Workshop Synthesis Syst. Integr. Mixed Inform. Technol.*, 2010, pp. 115–120.
[17] T. F. Chan, J. Cong, J. Shinnerl, K. Sze, and M. Xie, "mPL6: Enhanced multilevel mixed-size placement," in *Proc. ACM Int. Symp. Physical Design*, 2006, pp. 212–214.
[18] T. F. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proc. ACM Int. Symp. Phys. Design*, 2005, pp. 185–192.
[19] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 5, pp. 734–747, May 2005.
[20] A. B. Kahng and Q. Wang, "A faster implementation of APlace," in *Proc. ACM Int. Symp. Physical Design*, 2006, pp. 218–220.
[21] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design," U.S. Patent 6 370 673, 2002.
[22] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: Fast legalization of standard cell circuits with minimal movement," in *Proc. ACM Int. Symp. Phys. Design*, 2008, pp. 47–53.
[23] H. Zhou, "Efficient Steiner tree construction based on spanning graphs," in *Proc. ACM Int. Symp. Phys. Design*, 2003, pp. 152–157.
[24] ERLAB. (2000). *IBM-PLACE Benchmarks* [Online]. Available: http://er.cs.ucla.edu/benchmarks/ibm-place
[25] C. Albrecht. (2005). *IWLS 2005 Benchmarks* [Online]. Available: http://iwls.org/iwls2005/benchmarks.html
[26] Cadence Design Systems. *Encounter Digital Implementation System* [Online]. Available: http://www.cadence.com
[27] G.-J. Nam, "ISPD 2006 placement contest: Benchmarks suite and results," in *Proc. ACM Int. Symp. Phys. Design*, 2006, pp. 167–167.



Meng-Kai Hsu received the double B.S. degree in electronics engineering and computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2007, and the Ph.D. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2012.

His current research interests include large-scale mixed-size placement and 3-D integrated circuit placement.

Dr. Hsu is an Honorary Member of the Phi Tau Phi Scholastic Honor Society of Taiwan.

Valeriy Balabanov was born in Kyiv, Ukraine, in 1988. He received the B.S. degree in electro physics from National Chiao Tung University, Hsinchu, Taiwan, in 2009, and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2011, where he is currently pursuing the Ph.D. degree.

His current research interests include applications of mathematical logic, particularly in circuit design, formal verification, SAT, and QBF solving.

Yao-Wen Chang (S'94–A'96–M'96–SM'12–F'13) received the B.S. degree from National Taiwan University (NTU), Taipei, Taiwan, in 1988, and the M.S. and Ph.D. degrees from the University of Texas, Austin, in 1993 and 1996, respectively, all in computer science.

He is currently an Associate Dean of the College of Electrical Engineering and Computer Science, the Director of the Graduate Institute of Electronics Engineering, and a Distinguished Professor of the Department of Electrical Engineering, NTU. His current research interests lie in VLSI physical design and design for manufacturability. He has been working closely with industry in these areas. He has co-edited one textbook on EDA and co-authored one book on routing and over 210 ACM/IEEE conference/journal papers in these areas.

Dr. Chang is a 1st-Place Winner of three most recent contests, the 2012 ACM/IEEE DAC Placement Contest, the 2012 ACM ISPD Discrete Gate Sizing Contest, and the 2011 IEEE CEDA PATMOS Timing Analysis Contest. He is a five-time winner of the ACM ISPD contests on placement, global routing, clock network synthesis, and discrete gate sizing, and a recipient of six Best Paper Awards (ICCD, etc.) and 20 Best Paper Award nominations from DAC (five times), ICCAD (four times), etc. in the past ten years. He has received many research/teaching awards, such as the Distinguished Research Award (highest honor) from the National Science Council of Taiwan (twice), the IBM Faculty Awards, the CIEE Distinguished EE Professorship, the MXIC Young Chair Professorship, and excellent teaching awards from NTU (seven times). He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, and an editor of the IEEE DESIGN AND TEST OF COMPUTERS and the *Journal of Information Science and Engineering*. He has served as the General and Steering Committee Chairs of ISPD, and the Program Chair of ASP-DAC, FPT, and ISPD, and is on the IEEE CEDA Executive Committee, the ICCAD Executive Committee (Vice TPC Chair), the ASP-DAC Steering Committee, and the ACM/SIGDA Physical Design Technical Committee. He has served on the technical program committees of major EDA conferences, including ASP-DAC, DAC, DATE, FPL, FPT, GLSVLSI, ICCAD, ICCD, ISPD, SLIP, SOCC, and VLSI-DAT. He has served as an independent Board Director of Genesys Logic, a Technical Consultant of Faraday, MediaTek, and RealTek, the Chair of the EDA Consortium of the Ministry of Education, Taiwan, and a member of the Board of Governors of the Taiwan IC Design Society.