A New Multilevel Framework for Large-Scale Interconnect-Driven Floorplanning

Tung-Chieh Chen, *Student Member, IEEE*, Yao-Wen Chang, *Member, IEEE*, and Shyh-Chang Lin

***Abstract*—We present in this paper a new interconnect-driven multilevel floorplanner, called interconnect-driven multilevel-floorplanning framework (IMF), to handle large-scale building-module designs. Unlike the traditional multilevel framework that adopts the "Λ-shaped" framework (inaccurately called the "V-cycle" framework in the literature): bottom-up coarsening followed by top-down uncoarsening, the IMF, in contrast, works in the "V-shaped" manner: top-down uncoarsening (partitioning) followed by bottom-up coarsening (merging). The top-down partitioning stage iteratively partitions the floorplan region based on min-cut bipartitioning with exact net-weight modeling to reduce the number of global interconnections and, thus, the total wirelength. Then, the bottom-up merging stage iteratively applies fixed-outline floorplanning using simulated annealing for all regions and merges two neighboring regions recursively. Experimental results show that the IMF obtains the best published fixed-outline floorplanning results with the smallest average wirelength for the Microelectronics Center of North Carolina/ Gigascale Systems Research Center benchmarks. In particular, IMF scales very well as the circuit size increases. The V-shaped multilevel framework outperforms the Λ-shaped one in the optimization of global circuit effects, such as interconnection and crosstalk optimization, since the V-shaped framework considers the global configuration first and then processes down to local ones level by level, and thus, the global effects can be handled at earlier stages. The V-shaped multilevel framework is general and, thus, can be readily applied to other problems.**

***Index Terms*—Floorplanning, multilevel framework, partitioning, physical design, simulated annealing (SA).**

I. Introduction

As NANOMETER IC technologies advance, design complexity is growing at a dramatic speed. Modern chip designs often consist of millions of transistors, and designs with billions of transistors are already in production. To cope with the increasing design complexity, IP modules are widely reused for large-scale designs. Therefore, efficient and effective design methodology and tools capable of placing and optimizing large-scale modules are essential for modern chip designs.

A. Framework Evolution

The floorplanning frameworks are evolving to tackle the challenges with constantly increasing design complexity. Three major frameworks have been extensively studied in the literature: the flat, the hierarchical, and the multilevel frameworks. Many flat algorithms based on various floorplan representations have been proposed in the literature [2]–[10]. However, these algorithms do not scale well as the design size increases. To cope with the scalability problem, hierarchical approaches are proposed. The hierarchical approaches recursively divide a floorplanning region into a set of subregions and solve these subproblems independently. Adya *et al.* [11] propose a "floorplacement" framework (used in their program Capo 9) that combines partitioning and floorplanning techniques to handle both floorplanning and placement problems. It first partitions a floorplan and then finds legal subfloorplans. Cong *et al.* [12] present a fast floorplanner called PATOMA using look-ahead-enabled recursive bipartitioning. It partitions a floorplan and uses row-oriented block packing and zero-dead space floorplanning to find legal subfloorplans. Both the floorplacement and PATOMA are based on the hierarchical framework, in which the floorplanning stage is only used for legalization and overlap removal. The top-down hierarchical technique is efficient in handling large-scale problems. Nevertheless, a drawback of the hierarchical approaches is that they might lack the global information for the floorplanning interaction among different subregions.

To remedy the deficiency, the multilevel framework is proposed to solve the floorplanning problem, as well as graph/ circuit partitioning, placement, and routing. All of the existing multilevel frameworks adopt a two-stage technique, bottom-up coarsening followed by top-down uncoarsening, which is known as the "Λ-shaped" framework (note that this framework is often called the "V-cycle" framework in the literature; to differentiate from our new multilevel framework presented in this paper, nevertheless, we think it is more appropriate to name it the "Λ-shaped" framework as it works bottom-up and then top-down). Lee *et al.* [13] first proposed a Λ-shaped multilevel-floorplanning algorithm based on the B*-tree representation [2], called MB*-tree. It adopts a two-stage technique: clustering followed by declustering, which is based on the cost metric of area and connectivity. Hu *et al.* [14] proposed a Λ-shaped multilevel genetic floorplanning algorithm, called MLGFA. However, the algorithm only optimizes the chip area without considering the

Manuscript received September 25, 2005; revised October 7, 2006 and February 24, 2007. This work was supported in part by the Springsoft, Inc. and in part by the National Science Council of Taiwan under Grants NSC 93-2215-E-002-009, NSC 93-2220-E-002-001, and NSC 93-2752-E-002-008-PAE. This paper was presented in part at the 2005 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'05), November 2005. This paper was recommended by Associate Editor J. Lillis.

T.-C. Chen is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: tungchieh@ntu.edu.tw).

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

S.-C. Lin is with the R&D Center, Springsoft, Inc., Hsinchu 300, Taiwan, R.O.C. (e-mail: chris@springsoft.com.tw).

Digital Object Identifier 10.1109/TCAD.2007.907065

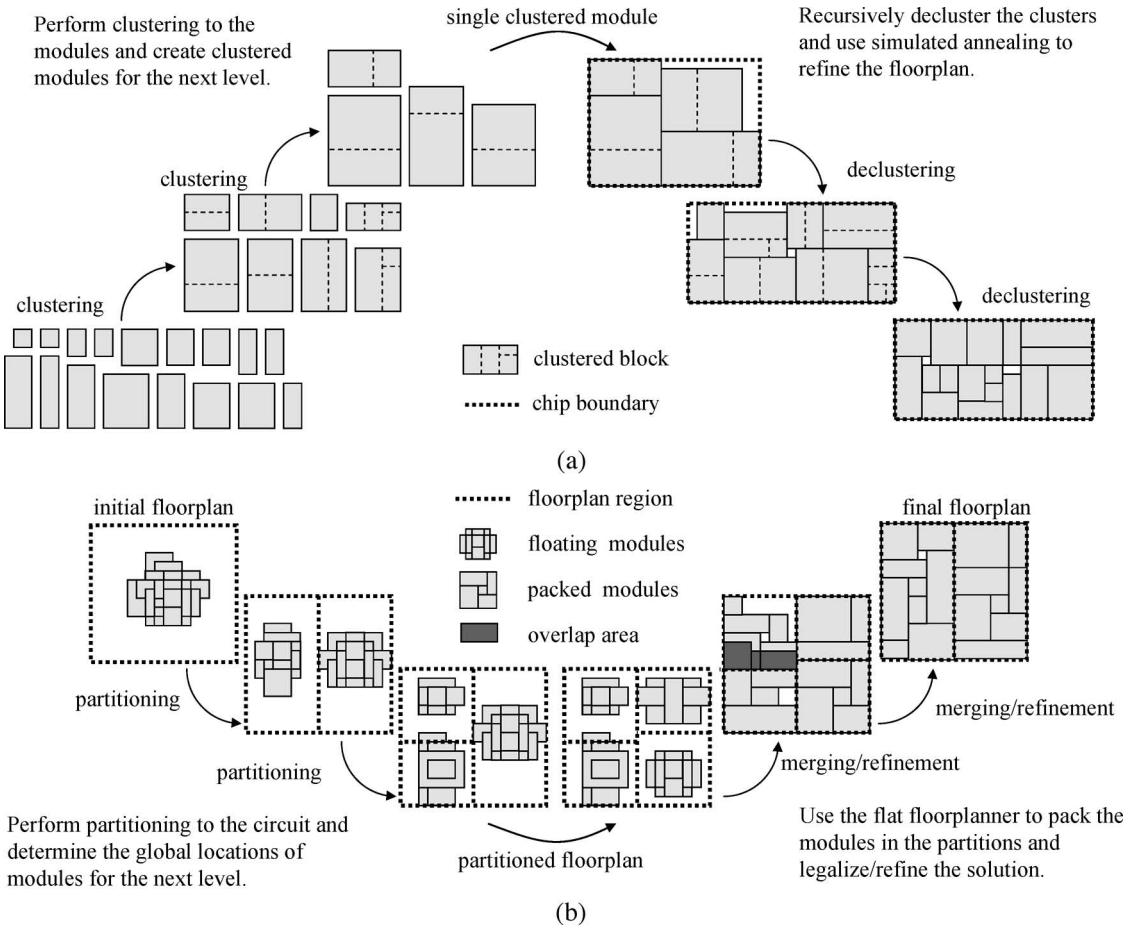

Fig. 1. (a) Λ-shaped multilevel framework of MB*-tree. (b) V-shaped multilevel framework of IMF.

wirelength. Furthermore, both of the multilevel-floorplanning algorithms consider only variable-die floorplanning.

B. Our Contributions

In this paper, we present the first "V-shaped" multilevel framework (note that this "V-shaped" multilevel framework is different from the "V-cycle" multilevel framework in the literature; as aforementioned, the so-called "V-cycle" multilevel framework is in fact the Λ-shaped one). Unlike the traditional Λ-shaped multilevel frameworks that apply the bottom-up coarsening followed by the top-down uncoarsening, our V-shaped multilevel framework adopts the two-stage technique of top-down uncoarsening followed by bottom-up coarsening. The V-shaped multilevel framework outperforms the Λ-shaped one in the optimization of global circuit effects, since the V-shaped framework first considers the global configuration and then processes down to local ones level by level, and thus, the global effects can be handled at earlier stages.

Based on the new framework, we develop the first V-shaped multilevel-floorplanning algorithm to handle the interconnect-driven large-scale floorplan designs. Our V-shaped interconnect-driven multilevel-floorplanning framework (IMF) adopts the two-stage technique of top-down partitioning followed by bottom-up merging. The top-down partitioning stage iteratively partitions the floorplan region based on min-cut bipartitioning with exact net-weight modeling (ENW) to reduce the number of global interconnections and, thus, the total wirelength. Then, the bottom-up merging stage iteratively applies fixed-outline floorplanning using simulated annealing (SA) for all regions and merges two neighboring regions recursively. Experimental results show that the IMF obtains the best fixed-outline floorplanning results with the smallest average wirelength for the Microelectronics Center of North Carolina (MCNC)/Gigascale Systems Research Center (GSRC) benchmarks, compared with the publicly available floorplanners B*-tree, MB*-tree, Parquet 3.1/4.0, and Capo 9.0/10.2. We also show that IMF scales very well as the circuit size increases, based on the `ami49_x` large-scale building-module benchmarks.

The remainder of this paper is organized as follows. Section II compares our V-shaped multilevel framework with the Λ-shaped and the hierarchical ones. Section III presents our floorplanning algorithms. Section IV shows the experimental results, and finally, the conclusions are given in Section V.

II. Multilevel Framework

The traditional Λ-shaped multilevel frameworks apply a two-stage technique: bottom-up coarsening followed by top-down uncoarsening. We take MB*-tree [13] for an example. Fig. 1(a) shows the MB*-tree multilevel framework. It is based on a two-stage technique of bottom-up clustering followed by top-down declustering. The clustering stage iteratively groups a set

of modules based on a cost metric of area and module connectivity. The declustering stage iteratively ungroups a set of the previously clustered modules and uses SA to refine the solution. Experimental results showed that MB*-tree obtains solutions of very small white space. For modern floorplanning, however, the interconnections among modules are a very important cost metric for routability and performance optimization and, thus, should be carefully considered. Since MB*-tree first works in a bottom-up manner by clustering local modules based on area and local connectivity, it does not have the view for the global configuration at the earlier stages. Therefore, it is very likely that MB*-tree can only obtain a suboptimal solution, since it may make a wrong choice during the clustering stage, and it may become very hard to further refine the floorplan solution during the declustering stage.

Fig. 1(b) illustrates our V-shaped IMF framework. Unlike MB*-tree that adopts a Λ-shaped framework, our IMF uses the V-shape of top-down partitioning followed by bottom-up merging. Partitioning determines the global locations of modules, while merging legalizes and refines the floorplan. At the initial step, all the modules are located at the center of the chip. Then, the chip is partitioned into two subregions, and modules are moved from the chip center to the centers of new subregions. The partitioning continues until the number of modules in a region is smaller than a threshold. Then, we obtain a "partitioned floorplan," and the partitioning stage finishes.

At the merging stage, fixed-outline floorplanning is applied to every region. We use SA to find a feasible floorplan to fit all modules into the region and minimize the wirelength. Then, two neighboring regions are merged into one larger region, and fixed-outline floorplanning is used again to refine the floorplan. Merging and refining are performed recursively until all regions are merged into one single top-level floorplan. Finally, the final floorplan result is obtained. Note that the IMF framework is independent of the floorplan representation, so every existing floorplan representation can be used.

Table I lists the characteristics of our IMF multilevel framework, the MB*-tree multilevel framework [13], the Capo floorplacement framework [11], and the PATOMA framework [12]. Our IMF framework and the MB*-tree framework are based on the multilevel framework, while the Capo framework and the PATOMA framework are based on the hierarchical framework. Although Capo and PATOMA use partitioning, unlike IMF, they do not have the refinement stage to further improve their results.

TABLE I
FRAMEWORK COMPARISONS

Framework	Characteristics
Our IMF multilevel framework	• Use the V-shaped multilevel framework. • Use top-down partitioning followed by bottom-up merging. • Handle fixed-die constraints. • Minimize the wirelength under the given area constraint.
The MB*-tree multilevel framework in [13]	• Use the Λ-shaped multilevel framework. • Use bottom-up clustering followed by top-down declustering. • Deal with variable dies. • Need to specify the weights for area and wirelength by the user.
The Capo floorplacement framework in [11]	• Use the top-down hierarchical framework. • Use top-down partitioning and fixed-outline floorplanning. • Handle fixed-die constraints. • Minimize the wirelength under the given area constraint.
The PATOMA framework in [12]	• Use the top-down hierarchical framework. • Use top-down partitioning and ZDS/ROB fast look-ahead floorplanning. • Handle fixed-die constraints. • Minimize the wirelength under the given area constraint.

III. ALGORITHM

The IMF algorithm consists of three steps: 1) chip-dimension determination; 2) the partitioning stage; and 3) the merging stage.

A. Chip-Dimension Determination

Given a set of modules with the total area A and the maximum white-space fraction γ, we can construct a fixed outline with the aspect ratio (height/width) α. The chip dimension (H^*, W^*) can be computed by $H^* = \sqrt{(1+\gamma)A\alpha}$ and $W^* = \sqrt{(1+\gamma)A/\alpha}$ [15]. If $\max(H_i, W_i)$ of a module m_i is larger than $\max(H^*, W^*)$, the module m_i can never fit into the chip boundary. In this case, the chip dimension can be computed by $H^* = \max(W_i, H_i)$ and $W^* = (1+\gamma)A/H^*$. The new dimension of the chip can ensure that every module fits into the chip boundary. Note that the chip area $A^* = H^*W^* = (1+\gamma)A$ remains the same as the original formulation.

B. Partitioning Stage

At the initial level, the locations of all modules are set to the center of the chip region. To prevent generating subregions of large aspect ratios, we choose the longer side to divide the region into two subregions. After the shapes of two subregions are determined, we move the modules to the two centers of the two subregions to minimize the half-perimeter wirelength (HPWL).

The module-location determination problem can be formulated as a hypergraph-partitioning problem. We first derive an ENW to map the HPWL cost exactly to the min-cut cost. With the exact modeling, in other words, minimizing HPWL is equivalent to finding the min-cut cost. Therefore, the given hypergraph is partitioned using a min-cut bipartitioner to obtain the minimum HPWL. The new locations of the modules are thus determined by the partitioner, and each subpartition corresponds to a subregion.

1) Exact Net-Weight Modeling: Since the net weight in the traditional terminal propagation (TTP) for the min-cut-based placement is a constant value, the weight with the change in HPWL cannot be exactly modeled, whether a net is cut or not. The underlying idea for our ENW (terminal propagation) is that we want to map the min-cut cost exactly to the HPWL change, which is similar to the "bounding-box-aware terminal propagation (BBTP)" proposed by Selvakkumaran and Karypis in [16] and [17]. They first proposed BBTP in [16] and, later, discussed the BBTP in detail under seven cases in [17]. Another net-weighting method was proposed in [18], in which the

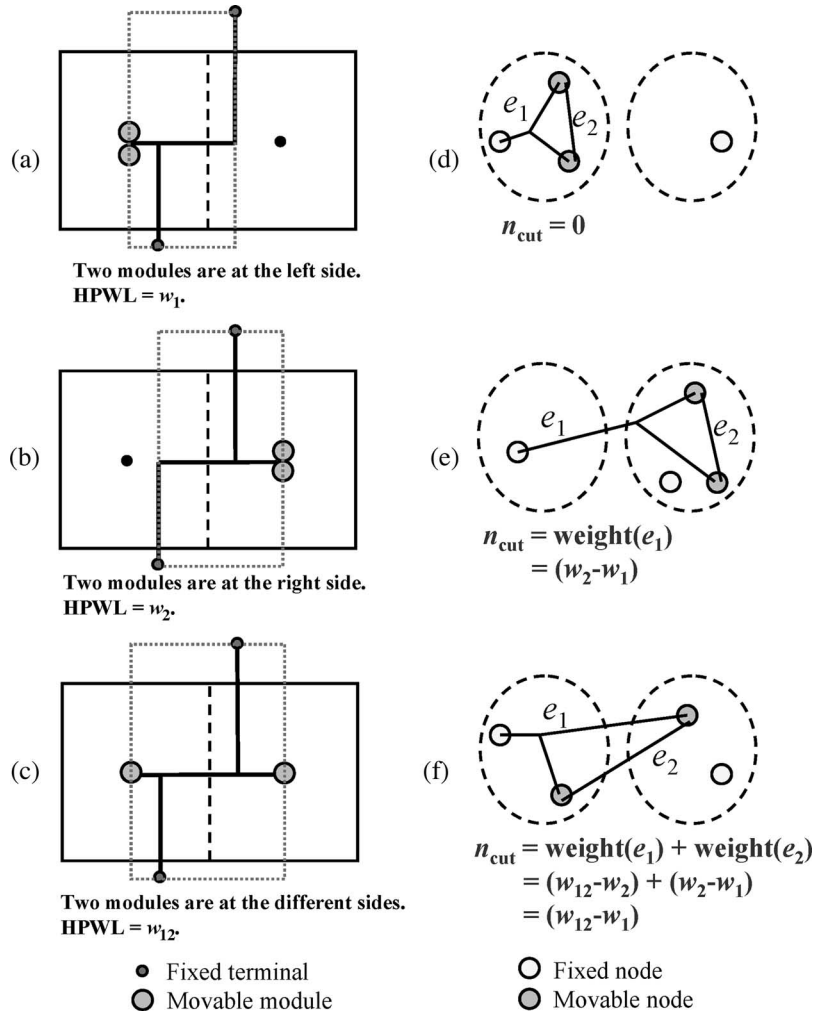

Fig. 2. Example of determining a net weight. (a)–(c) Three possible partitioning results. (d)–(f) Corresponding partitioning hypergraphs.

authors discussed the net-weighting method for partitioning based on four cases. However, they can obtain exact modeling only for two-terminal nets, i.e., they can only obtain suboptimal results for multiterminal nets. Unlike the previous work that exhaustively enumerates potential cases, we derive a unified model to assign the net weights to map the HPWL value exactly. Our HPWL modeling can not only be applied to vertical-cut or horizontal-cut partitioning but can also be applied to placement feedback (repartitioning) [19]. Furthermore, our unified HPWL model can even apply to the partitioning associated with two nonadjacent regions, for which the method presented in [17] would need to enumerate tens of cases for the HPWL modeling and thus is obviously much more complex and harder for implementation.

We give our unified HPWL modeling as follows. A circuit is modeled as a hypergraph. Each node in the hypergraph corresponds to a module inside the target region, with the node weight being set to the area of the corresponding module. Each hyperedge denotes a two- or multiterminal net in the circuit, with the hyperedge weight being set to the value of the HPWL contribution if the hyperedge is cut.

Let w_1 be the HPWL when all modules are at the side closer to the span of the terminals; w_2 is the HPWL when all modules are at the opposite side, and w_{12} is the HPWL when modules are at the both sides (see Fig. 2 for an illustration). Let $n_{\rm cut}$ be the cutsize of the net for the corresponding hypergraph. Therefore, we have $w_{12} \geq w_2 \geq w_1$. Then, we introduce a partitioning hypergraph with two fixed nodes to represent the two sides and with movable nodes to represent the movable modules. We then add two hyperedges e_1 and e_2 into the hypergraph. The hyperedge weight can be determined as follows. We introduce e_1 to connect the fixed node corresponding to the side closer to the span of terminals and all movable nodes and e_2 to connect between all movable nodes. We then assign the weight of the hyperedge e_1 as the value $w_2 - w_1$ (note that $w_2 \geq w_1$) and that of the hyperedge e_2 as the value $w_{12} - w_2$. Partitioning the resulting hypergraph can determine to which partition the module belongs. We have the following theorem.

Theorem 1: With the unified net-weight modeling, we have HPWL $= w_1 + n_{\rm cut}$.

Proof: There are three possible partitioning results: 1) All nodes are at the partition corresponding to the side closer to the span of terminals; 2) all nodes are at the other partition; and 3) nodes are at the two different partitions. Without loss of generality, we use Fig. 2(d)–(f) to represent respective cases 1)–3), and the three partitioning results correspond to

the configurations shown in Fig. 2(a)–(c), respectively. For easier explanation, a four-terminal net with two fixed terminals (representing the two sides of the partitions) and two modules is considered (nevertheless, the following claims still hold for other cases). The x-range of the two terminals (i.e., the span between the two terminals in the x-direction) is within that of the centers of the two partitions, and the center of the left partition is closer to the x-range. For each net, we compute three HPWL values (w_1, w_2, and w_{12}). In Fig. 2(a), the two modules are at the left side. w_1 is equal to the half of the bounding box shown in Fig. 2(a), represented by the dotted lines. w_2 is the HPWL when the two modules are at the opposite side [the right side for the example shown in Fig. 2(b)]. Similarly, w_{12} is the HPWL when the two modules are at different sides [Fig. 2(c)]. For the case shown in Fig. 2, it is easy to see that $w_{12} \geq w_2 \geq w_1$. For the case of Fig. 2(d), no hyperedge in the resulting hypergraph is cut. Therefore, its cutsize $n_{\text{cut}} = 0$. In Fig. 2(e), e_1 is cut, and the cutsize is given by $n_{\text{cut}} = \text{weight}(e_1) = w_2 - w_1$. In Fig. 2(f), both e_1 and e_2 are cut, and thus, the cutsize $n_{\text{cut}} = \text{weight}(e_1) - \text{weight}(e_2) = (w_{12} - w_2) + (w_2 - w_1) = w_{12} - w_1$. For all of these three cases, we conclude that the corresponding HPWL (which are w_1, w_2, and w_{12}, respectively) is given by $w_1 + n_{\text{cut}}$ ($w_1 + 0, w_1 + (w_2 - w_1)$, and $w_1 + (w_{12} - w_1)$, respectively). The claims for the cases with a net of more than two terminals are similar. ■

Furthermore, we have the following theorem.

Theorem 2: The unified net-weight modeling exactly maps HPWL to the min-cut cost.

Proof: Let HPWL_i be the HPWL of net i, and $w_{1,i}$ be the HPWL of net i when its modules are all at the side closer to the span of the terminals [see Fig. 2(a)] and $n_{\text{cut},i}$ be the cutsize of net i. By Theorem 1, we have

$$\begin{aligned}\min\left(\sum \text{HPWL}_i\right) &= \min\left(\sum (w_{1,i} + n_{\text{cut},i})\right) \\ &= \sum w_{1,i} + \min\left(\sum n_{\text{cut},i}\right)\end{aligned}$$

since $\sum w_{1,i}$ is a constant. Therefore, the unified net-weight modeling exactly maps HPWL to the min-cut cost. Thus, finding the minimum HPWL is equivalent to finding the min-cut as long as the external terminals are given. ■

The partitioning stage continues until the number of modules in each partition is smaller than a threshold. Then, the partitioned floorplan is obtained.

C. Merging Stage

In the merging stage, we first use fixed-outline floorplanning to pack the modules in the partition and, then, merge two neighboring regions into one larger region if the merging leads to a better result. The fixed-outline floorplanning is applied again to legalize/refine the floorplan. If the merging does not result in a better solution, we will simply keep the original floorplan.

1) Fixed-Outline Floorplanning: Each region has its own height and width, and all modules in the region must fit into the region to generate a feasible floorplan. We treat the modules and I/O pads outside the current region as fixed terminals. We use the B*-tree representation with SA to find a feasible floorplan within the region. The reasons are twofold: 1) The B*-tree has been shown an efficient and effective data structure for floorplan design [2]; and 2) we intend to make fair comparison with the state-of-the-art multilevel-floorplanning work MB*-tree [13], which is also based on the B*-tree.

The cost function Φ for SA is similar to the one in [20], and it is defined as follows:

$$\Phi = k_1 \frac{A_F}{A_{F,\text{norm}}} + k_2 \frac{W_L}{W_{L,\text{norm}}} + k_3 \left(\frac{W_F}{H_F} - \frac{W_R}{H_R}\right)^2 \quad (1)$$

where A_F is the current floorplan area, $A_{F,\text{norm}}$ is the area normalization factor, W_L is the current wirelength, $W_{L,\text{norm}}$ is the wirelength normalization factor, W_F is the current floorplan width, H_F is the current floorplan height, W_R is the width of the region, H_R is the height of the region, and k_1, k_2, and k_3 are user-specified parameters. To calculate the area/wirelength normalization factors, several random perturbations are performed before SA starts, and $A_{F,\text{norm}}(W_{L,\text{norm}})$ is set to the average value of $A_F(W_L)$.

2) Accelerative Fixed-Outline Floorplanning (AFF): We observe that it takes much more time for fixed-outline floorplanning than for partitioning/merging. Therefore, we propose an AFF technique to speed up the whole framework. Typically, fixed-outline floorplanning spends most time in computing the wirelength, as also observed in [21]. Furthermore, many floorplanning results might not be feasible because the resulting floorplans cannot fit into the bounding box. To speed up floorplanning, we first set $k_2 = 0$ for the cost function to perform area-driven fixed-outline floorplanning. Then, we calculate the wirelength only when the floorplan can fit into the bounding box with a smaller cost. If the resulting wirelength is better than the best wirelength, we save the current result as the best result. Since we can reduce significant running time for computing the wirelength, we may increase the number of perturbations to search for better floorplan solutions. This technique can reduce significant running time without trading too much solution quality.

3) Partition Merging: If the fixed-outline floorplanning cannot find a feasible floorplan within the bounding box, then we still keep the solution. In the next refinement level, two partitioned regions are merged. To merge two vertical regions, we make the root of the B*-tree for the upper subfloorplan as the right child of the right-most node of the B*-tree for the bottom subfloorplan. The width of the merged floorplan is equal to the maximum width of the subfloorplans, and the height of the floorplan is less than or equal to the sum of the two subfloorplan's heights due to the packing. To merge two horizontal regions, we first find the node which corresponds to the right-most module of the left subfloorplan. Then, we make the root of the B*-tree for the right subfloorplan as the left child of the node we found. The height of the merged floorplan is equal to the maximum height of the two subfloorplans, and the width of the merged floorplan is equal to the sum of the two subfloorplan's widths.

Interconnect-Driven Multilevel Floorplanning (IMF)
Input: Modules, nets, I/O pads, fixed-die parameters. Output: A feasible floorplan within the fixed-die with wirelength (HPWL) being minimized.
1. Initialize the data structures and the chip dimension; 2. Set all modules at the center of the floorplan region; 3. **The Partitioning Stage**: 4. **until** every region has fewer than n_{max} modules 5. Choose a partition; 6. Create a hypergraph model; 7. Bipartition the hypergraph; 8. Move modules to the new sub-regions; 9. Generate the partitioned floorplan; 10. **The Merging Stage**: 11. **while** there exists more than one region 12. Choose two neighboring regions; 13. Apply fixed-outline floorplanning; 14. Merge two regions; 15. Generate the final floorplan; 16. **return** the final floorplan;

Fig. 3. IMF algorithm.

The merging stage iteratively merges two previously partitioned regions and then refines the floorplan solution based on fixed-outline SA. The merging stage continues until all regions are merged into one top-most region, and the final floorplan is obtained.

D. Algorithm

Fig. 3 summarizes our algorithm. The inputs are dimensions of modules, a (multiterminal) netlist, the location of the I/O pads, and fixed-die parameters. We first initialize our data structures and determine the chip boundary. Then, all modules are set to the center of the floorplanning region. In the partitioning stage, we create a hypergraph for the current region and apply a state-of-the-art hypergraph/circuit partitioner, such as hMetis [22], to obtain a min-cut bipartitioning result (by using hMetis which is based on the Λ-shaped multilevel framework, our implementation applies the V-shaped framework with the Λ-shaped one inside the partitioning stage). The modules are then moved to the centers of the subregions according to the partitioning result. The partitioning stage continues until every region has fewer modules than $n_{\max}$ modules, and the partitioned floorplan is obtained. In the merging stage, the fixed-outline floorplanning with wirelength minimization is applied to pack the modules into the regions. Then, two regions are merged into a larger one if the merging leads to a better result; otherwise, we will keep the original floorplan. After all regions are merged, we obtain the final floorplan.

Note that our V-shaped multilevel-floorplanning framework is different from the floorplacement framework [11] as follows. 1) The floorplacement framework calls the floorplanner during the partitioning stage when there exists large blocks or many blocks in the region. After floorplanning, large blocks are fixed, and the partitioning process continues with the remaining blocks. In our V-shaped framework, we do not fix any blocks during the partitioning stage, and thus, we have more flexibility in determining the block positions. 2) If the floorplanner cannot find a feasible solution in the floorplacement framework, it only merges the neighboring regions and refines the merged region once. In contrast, our V-shaped multilevel framework recursively merges and refines the regions up to the top level, which has higher chances to find a desirable solution.

TABLE II
COMPARISONS OF THE HPWL BY USING THE TTP AND THE ENW. IN EACH ENTRY, BOTH THE MINIMUM/AVERAGE VALUES OBTAINED IN TEN RUNS ARE REPORTED

Circuit	TTP	ENW	TTP/ENW Ratio
n100	202713/203685	191199/191382	1.06/1.07
n200	365047/366104	341425/341660	1.07/1.07
n300	478947/479934	450806/453806	1.06/1.06
Average	–	–	1.06/1.07

IV. EXPERIMENTAL RESULTS

We made the comparisons with the following state-of-the-art floorplanning algorithms/packages: our IMF, B*-tree [2], MB*-tree [13], Parquet-3.1/-4.0 [23], PATOMA [12], Capo 9.0 [11], and Capo 10.2 [11] (note that Capo 10.2 was released in May 2006 when this paper was under revision). We used the MCNC and the GSRC [24] benchmark suites. All programs were compiled with gcc 3.3.2, and all experiments were preformed on a Linux PC with an Intel Pentium-4 3.2-GHz CPU with 3-GB memory. Note that, although, most existing standard-cell/mixed-size placers can handle the large-scale circuit-placement problem. They usually focus more on the standard-cells of the same height. Therefore, the standard-cell/mixed-size placers cannot handle large-scale building-module floorplanning well. We have tried well-known publicly available placers, such as Feng Shui 2.6/5.0 [26] and mGP [27], [28]. They all cannot obtain feasible or desirable building-module floorplans. Therefore, we shall not compare with those mixed-size placers.

For fair comparisons, we set the maximum white-space fraction γ to 15% and the chip aspect ratio α to 1% for all circuits. The I/O pads were scaled to the chip boundary. The wirelength was estimated using HPWL.

A. Exact Net-Weight Modeling

Table II compares the net-weight modeling based on the TTP and our ENW. For TTP, all net-weights are equal to 1.0. For ENW, the weights of the nets are assigned according to the scheme described in Section III-B1. The experiments were taken on the three largest GSRC benchmark for testing the effectiveness of the partitioning. We used the state-of-the-art Λ-shaped multilevel partitioner hMetis [22] (version 1.5.3) for min-cut partitioning. We performed partitioning on each circuit ten times to get the minimum/average HPWL. We set $n_{\max} = 10$ (i.e., partition all regions until each of them contains fewer than ten modules). From Table II, we observe that the ENW can reduce the HPWL by 6%–7%. Note that the runtimes are not reported here for both methods since they are about the same.

B. Comparisons of Solution Quality

Table III lists the HPWLs and CPU times obtained by B*-tree, MB*-tree, Capo, Parquet, PATOMA, IMF, and IMF with

TABLE III
COMPARISONS FOR HPWL AND CPU TIME AMONG B*-TREE, MB*-TREE, CAPO, PARQUET, PATOMA, IMF, AND IMF+AFF. * THE HPWL VALUES WITHIN THE PARENTHESES DENOTE THAT THE RESULTS CANNOT FIT INTO THE BOUNDING BOXES (B*-TREE/MB*-TREE) OR THAT THERE ARE OVERLAPS IN THE RESULT (CAPO 9/PATOMA). THE AVERAGE HPWL RATIO CONSIDERS ONLY THE RESULTS THAT CAN FIT INTO THE BOUNDING BOXES

			Variable-die floorplanner				Fixed-die floorplacer				Fixed-die floorplanner							
			B*-tree		MB*-tree		Capo 9.0		Capo 10.2		Parquet 3.1/4.0		PATOMA		**IMF**		**IMF+AFF**	
Circuit	#mod	#net	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU
apte	9	97	*(4.10 e5)	0.1	*(4.10 e5)	0.2	5.90 e5	0.2	5.04 e5	0.5	4.48 e5	0.2	5.42 e5	0.0	4.25 e5	0.7	5.00 e5	0.1
xerox	10	203	*(4.87 e5)	0.3	*(4.92 e5)	0.2	5.31 e5	1.2	6.30 e5	0.6	5.24 e5	0.6	NR	NR	5.05 e5	0.9	5.56 e5	0.2
hp	11	83	*(1.17 e5)	0.2	*(1.18 e5)	0.2	*(1.30 e5)	1.2	1.60 e5	0.9	1.30 e5	0.4	NR	NR	1.24 e5	0.7	1.75 e5	0.1
ami33	33	123	*(5.33 e4)	1.3	*(5.32 e4)	2.2	7.32 e4	3.4	6.15 e4	4.0	6.80 e4	1.7	NR	NR	6.20 e4	2.0	6.52 e4	0.8
ami49	49	408	*(7.64 e5)	5.7	*(7.23 e5)	3.9	1.00 e6	3.9	9.86 e5	5.2	8.83 e5	5.5	NR	NR	8.68 e5	5.4	9.48 e5	1.2
n10	10	118	*(3.48 e4)	0.1	*(3.50 e4)	0.2	4.19 e4	0.4	3.81 e4	0.4	3.80 e4	0.3	NR	NR	3.61 e4	0.5	3.92 e4	0.2
n30	30	349	*(1.05 e5)	1.2	*(1.06 e5)	2.5	1.10 e5	0.8	1.10 e5	1.0	1.14 e5	2.3	NR	NR	1.07 e5	1.7	1.08 e5	0.6
n50	50	485	*(1.32 e5)	3.9	*(1.36 e5)	9.3	1.41 e5	2.3	1.40 e5	3.6	1.47 e5	4.6	NR	NR	1.34 e5	7.0	1.36 e5	1.5
n100	100	885	*(2.04 e5)	21	*(2.14 e5)	37.5	2.24 e5	4.6	2.14 e5	5.5	2.42 e5	17.5	NR	NR	2.08 e5	11.5	2.09 e5	2.3
n200	200	1585	*(3.76 e5)	94.9	*(4.05 e5)	66.3	3.86 e5	15.0	3.72 e5	12.5	4.33 e5	77.2	4.56 e5	0.6	3.70 e5	64.6	3.73 e5	4.2
n300	300	1893	*(5.19 e5)	194.9	*(6.13 e5)	85.5	5.23 e5	13.5	5.00 e5	14.3	6.47 e5	166.2	6.94 e5	0.6	4.90 e5	91.7	4.94 e5	5.5
Comp.			*(0.96)	0.85	*(0.99)	1.01	1.11	0.69	1.09	0.76	1.10	1.04	1.31	0.02	1.00	1.00	1.09	0.23

(CPU is in seconds.)

AFF (IMF+AFF) for the MCNC/GSRC benchmarks. The number of modules ranges from 9 to 300, and the number of nets ranges from 83 to 1893. The average HPWL and CPU time are normalized to the result of IMF. Since B*-tree and MB*-tree cannot handle fixed-die constraints, we set their mode to wirelength optimization alone to make the comparisons. By doing so, B*-tree and MB*-tree should gain some advantages. In contrast, Parquet is a fixed-outline floorplanner (which is based on the sequence-pair representation). For fair comparison, the I/O pads for all circuits are fixed along the user-specified chip boundaries. Thus, the I/O pad locations for variable-die floorplanners are the same as those for fixed-die floorplanners. Furthermore, we reported the best results obtained by Parquet 3.1 and 4.0 since neither version dominated the other. We used the default wirelength minimization mode ($-$`minWL`) and set the maximum white-space fraction $\gamma = 0.15$ and chip aspect ratio 1.0, except that the aspect ratio of the circuit `hp` was set to 1.075 to allow all modules to fit into its bounding box. The average white space of the B*-tree (MB*-tree) variable-die floorplanner is about 24.1% (26.2%), and none of its resulting floorplans fits into the bounding box. The white spaces of the fixed-die floorplanners are all less than 15.0%, and all results can fit into the bounding boxes (however, the floorplacer Capo 9.0 does not legalize all modules for the benchmark circuit `hp`).

As shown in Table III, IMF reduces the HPWL by 10% (31%) on average as compared with Parquet (PATOMA). Although B*-tree (MB*-tree) minimizes the HPWL alone, it only reduces the HPWL by 4% (1%) and incurs significantly larger white spaces as compared with IMF. The CPU times for B*-tree and Parquet are comparable, while the multilevel floorplanners, MB*-tree and IMF, and the hierarchical floorplacer/floorplanner Capo and PATOMA spend much less CPU time, particularly for larger circuits. IMF+AFF achieves 4.3× speedup at the cost of 9% HPWL overhead as compared with IMF. Based on the results, the AFF can significantly reduce the running time, and it is particularly suitable for large-scale circuits since it achieves larger speedup (e.g., 16.7× for `n300`) with only 1% overhead in the HPWL value. Therefore, the AFF option is turned on when handling large-scale circuits. The runtime of the Capo floorplacer is between that of the IMF and the IMF+AFF. The solution quality of Capo 9.0 is 11% and 2% worse than the IMF's and the IMF+AFF's solution quality, respectively. The solution quality of Capo 10.2 is 9% worse than the IMF's solution quality and similar to the IMF+AFF's solution quality, but its runtime is larger than the IMF+AFF's runtime. PATOMA is very fast, but it fails to find legal floorplans for many circuits, and its resulting HPWL is not good enough.

C. Scalability of the Floorplanners

To verify the scalability of the algorithms/floorplanners, we tested on the large-scale floorplan-benchmark circuits used in [13], which are duplicated from the largest MCNC benchmark circuit `ami49`. For [13], they simply duplicated all modules and nets of the circuit `ami49`. However, these kinds of synthetic circuits are not general, since there is no interconnection between the duplicated copies of circuits. The clustering (partitioning) method would take advantage of these kinds of special structures. To make the comparison fairer, we also added interconnections among different copies of duplicated circuits. For the circuit `ami49_x`, we duplicated each module/net x times. For each module m_i, we duplicated it as $m_{i,1}, m_{i,2}, \ldots, m_{i,x}$ and added $x-1$ nets between $(m_{i,1}, m_{i,2})$, $(m_{i,1}, m_{i,3}), \ldots, (m_{i,1}, m_{i,x})$. We divide block widths/heights by five for the benchmarks to avoid overflows in computing the wirelength.

Table IV lists the results for the five algorithms/floorplanners on `ami49_x` circuits. The average HPWL ratio is normalized to that of the IMF+AFF. The resulting HPWL of the IMF+AFF on average outperforms B*-tree, MB*-tree, Parquet, PATOMA, and Capo 9.2 by about 20%, 29%, 56%, 36%, and 5%, respectively. Capo 10.2 obtains 2% shorter HPWL than the IMF+AFF's but uses 3.23× CPU time on average. The resulting HPWL ratios for Parquet and MB*-tree grow up to 50% or more as the circuit size increases. The CPU times for B*-tree and Parquet grow dramatically as the circuit size increases while both the IMF+AFF and Capo can scale to very large-scale designs. The experimental

TABLE IV
COMPARISONS FOR HPWL AND CPU TIME AMONG B*-TREE, MB*-TREE, CAPO, PARQUET, PATOMA, AND MF+AFF FOR ami49_x CIRCUITS. NR: NO LEGAL RESULT OBTAINED (PATOMA) OR CPU TIME EXCEEDED 12-HR (B*-TREE/PARQUET)

			Variable-die floorplanner				Fixed-die floorplacer				Fixed-die floorplanner					
			B*-tree		MB*-tree		Capo 9.0		Capo 10.2		Parquet 3.1/4.0		PATOMA		**IMF+AFF**	
Circuit	#mod	#net	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU	HPWL	CPU
ami49_1	49	408	1.60 e5	0.1	1.49 e5	0.1	1.96 e5	0.0	1.78 e5	0.1	1.88 e5	0.1	NR	NR	1.85 e5	0.0
ami49_2	98	865	4.59 e5	0.4	3.88 e5	0.4	5.36 e5	0.1	5.05 e5	0.2	5.38 e5	0.3	NR	NR	4.67 e5	0.0
ami49_4	196	1779	1.16 e6	2	1.06 e6	0.8	1.14 e6	0.3	1.06 e6	0.5	1.56 e6	1.3	1.58 e6	0.0	1.07 e6	0.1
ami49_10	490	4521	4.28 e6	16.9	3.79 e6	2.1	3.48 e6	0.7	3.20 e6	0.9	6.03 e6	9.6	4.97 e6	0.0	3.31 e6	0.2
ami49_20	980	9091	1.18 e7	94.1	1.02 e7	4.8	8.48 e6	1.5	7.66 e6	1.9	1.61 e7	44.5	1.16 e7	0.1	8.13 e6	0.5
ami49_40	1960	18231	3.08 e7	609.1	2.67 e7	12.7	2.07 e7	3.3	1.93 e7	3.7	4.04 e7	262.9	2.70 e7	0.1	2.02 e7	1.3
ami49_60	2940	27371	NR	NR	4.99 e7	25.7	3.54 e7	5.5	3.37 e7	6.4	7.41 e7	734.4	4.65 e7	0.2	3.42 e7	2.2
ami49_80	3920	36511	NR	NR	7.48 e7	36.7	5.22 e7	8.8	4.95 e7	7.8	NR	NR	6.65 e7	0.4	5.05 e7	3.5
ami49_100	4900	45651	NR	NR	1.03 e8	66.2	7.13 e7	10.6	6.79 e7	10.4	NR	NR	8.85 e7	0.5	6.86 e7	5.1
ami49_150	7350	68501	NR	NR	1.84 e8	107.5	1.24 e8	13.7	1.19 e8	18.4	NR	NR	1.53 e8	1.0	1.21 e8	11.0
ami49_200	9800	91351	NR	NR	3.34 e8	224.8	1.83 e8	19.8	1.77 e8	22.4	NR	NR	2.28 e8	1.7	1.80 e8	19.1
Comp.			1.20	190.31	1.29	10.48	1.05	2.53	0.98	3.23	1.56	100.60	1.36	0.10	1.00	1.00

CPU is in minutes.

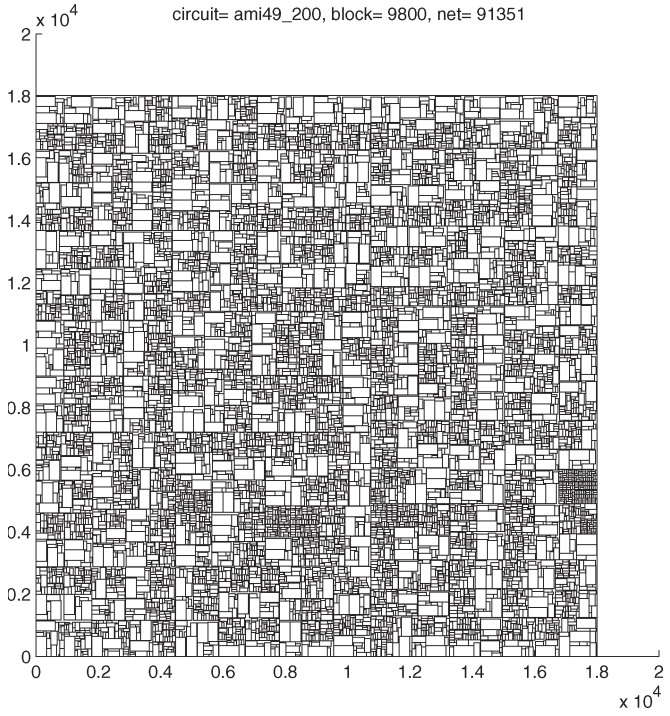

Fig. 4. Resulting floorplan for ami49_200 using the IMF+AFF.

results show that the IMF+AFF has superior scalability and maintains high-quality results for large-scale designs. Fig. 4 shows the resulting layout of the circuit ami49_200 by using the IMF+AFF.

V. CONCLUSION

We have presented a new IMF algorithm based on the V-shaped multilevel framework. IMF adopts a two-stage technique: partitioning followed by merging. The ENW is used in the partitioning stage, and an AFF is applied in the merging stage. Experimental results show that the IMF+AFF scales very well as the circuit size and interconnection complexity increase. The V-shaped multilevel framework is general and, thus, can be applied to other problems.

REFERENCES

[1] T.-C. Chen, Y.-W. Chang, and S.-C. Lin, "IMF: Interconnect-driven multilevel floorplanning for large-scale building-module designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2005, pp. 159–164.

[2] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proc. ACM/IEEE Des. Autom. Conf.*, Los Angeles, CA, Jun. 2000, pp. 458–463.

[3] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," in *Proc. ACM/IEEE Des. Autom. Conf.*, New Orleans, LA, Jun. 1999, pp. 268–273.

[4] X. Hong, G. Huang, T. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, "Corner block list: An effective and efficient topological representation of non-slicing floorplan," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2000, pp. 8–12.

[5] J.-M. Lin and Y.-W. Chang, "TCG: A transitive closure graph-based representation for non-slicing floorplans," in *Proc. ACM/IEEE Des. Autom. Conf.*, Las Vegas, NV, Jun. 2001, pp. 764–769.

[6] J.-M. Lin and Y.-W. Chang, "TCG-S: Orthogonal coupling of P*-admissible representations for general floorplans," in *Proc. ACM/IEEE Des. Autom. Conf.*, New Orleans, LA, Jun. 2002, pp. 842–847.

[7] J.-M. Lin, Y.-W. Chang, and S.-P. Lin, "Corner sequence—A P-admissible floorplan representation with a worst case linear-time packing scheme," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 4, pp. 679–686, Aug. 2003.

[8] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajatani, "Rectangle-packing based module placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 1995, pp. 472–479.

[9] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajatani, "Module placement on bsg-structure and IC layout applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 1996, pp. 484–491.

[10] E. F. Y. Young, C. C. N. Chu, and Z. C. Shen, "Twin binary sequences: A nonredundant representation for general nonslicing floorplan," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 4, pp. 457–469, Apr. 2003.

[11] 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.*, San Jose, CA, Nov. 2004, pp. 550–557.

[12] J. Cong, M. Romes, and J. R. Shinnerl, "Fast floorplanning by look-ahead enabled recursive bipartitioning," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Shanghai, China, Jan. 2005, pp. 1119–1122.

[13] H.-C. Lee, Y.-W. Chang, J.-M. Hsu, and H. H. Yang, "Multilevel floorplanning/placement for large-scale modules using B*-trees," in *Proc. ACM/IEEE Des. Autom. Conf.*, Anaheim, CA, Jun. 2003, pp. 812–817.

[14] C.-C. Hu, D.-S. Chen, and Y.-W. Wang, "Fast multilevel floorplanning for large scale modules," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 2004, pp. 205–208.

[15] S. N. Adya and I. Markov, "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 1120–1135, Dec. 2003.

[16] N. Selvakkumaran and G. Karypis, "Theto—A fast and high-quality partitioning driven global placer," Dept. Comput. Sci. Eng., Univ. Minnesota, Minneapolis, MN, Tech. Rep. 03-46, Nov. 2003.

[17] N. Selvakkumaran and G. Karypis, "Theto—A fast and high-quality partitioning driven placement tool," Dept. Comput. Sci. Eng., Univ. Minnesota, Minneapolis, MN, Tech. Rep. 04-40, Oct. 2004.
[18] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang, and Y.-W. Chang, "NTUplace: A ratio partitioning based placement algorithm for large-scale mixed-size designs," in *Proc. ACM Int. Symp. Phys. Des.*, San Francisco, CA, Apr. 2005, pp. 236–238.
[19] A. B. Kahng and S. Reda, "Placement feedback: A concept and method for better min-cut placements," in *Proc. ACM/IEEE Des. Autom. Conf.*, San Diego, CA, Jun. 2004, pp. 357–362.
[20] T.-C. Chen and Y.-W. Chang, "Modern floorplanning based on fast simulated annealing," in *Proc. ACM Int. Symp. Phys. Des.*, San Francisco, CA, Apr. 2005, pp. 104–112.
[21] H. H. Chan, S. N. Adya, and I. L. Markov, "Are floorplan representations important in digital design?" in *Proc. ACM Int. Symp. Phys. Des.*, San Francisco, CA, Apr. 2005, pp. 129–136.
[22] hMetis. [Online]. Available: http://www-users.cs.umn.edu/~karypis/metis/hmetis/
[23] PARQUET. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/parquet/
[24] 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.
[25] *GSRC Floorplan Benchmarks*. [Online]. Available: http://www.cse.ucsc.edu/research/surf/GSRC/progress.html
[26] *FengShui Placer*. [Online]. Available: http://vlsicad.cs.binghamton.edu/software.html
[27] *mGP: Multilevel Global Placement*. [Online]. Available: http://ballade.cs.ucla.edu/mGP/
[28] C.-C. Chang, J. Cong, and X. Yuan, "Multi-level placement for large-scale mixed-size IC designs," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Kitakyushu, Japan, Jan. 2003, pp. 325–330.

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

His current research interests include very large-scale integration floorplanning and placement.

Yao-Wen Chang (S'94–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, 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 manufacturing, and FPGA. He has been working closely with industry on projects in these areas. He has coauthored one book on routing and over 120 ACM/IEEE conference/journal papers in these areas.

Dr. Chang received an award at the 2006 ACM ISPD Placement Contest, Best Paper Award at ICCD-1995, and nine Best Paper Award Nominations from DAC-2007, ISPD-2007 (two), DAC-2005, 2004 ACM TODAES, ASP-DAC-2003, ICCAD-2002, ICCD-2001, and DAC-2000. He has received many awards for research performance, such as the inaugural First-Class Principal Investigator Awards and the 2004 Mr. Wu Ta You Memorial Award from the National Science Council of Taiwan, the 2004 MXIC Young Chair Professorship from the MXIC Corp, and for excellent teaching from National Taiwan University and National Chiao Tung University. He is an editor of the Journal of Computer and Information Science. He has served on the ACM/SIGDA Physical Design Technical Committee and the technical program committees of ASP-DAC (topic chair), DAC, DATE, FPT (program co-chair), GLSVLSI, ICCAD, ICCD, IECON (topic chair), ISPD, SOCC (topic chair), TENCON, and VLSI-DAT (topic chair). He is currently an independent board member of Genesys Logic, Inc, the chair of the Design Automation and Testing (DAT) Consortium of the Ministry of Education, Taiwan, a member of the board of governors of the Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.

Shyh-Chang Lin received the B.S. degree in control engineering from the National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1989 and the M.S. and Ph.D. degrees in electrical engineering from Michigan State University, East Lansing, in 1993 and 1997, respectively.

He is currently with the R&D Center, Springsoft, Inc., Hsinchu. His research interests include analog-layout automation and physical-design automation for very large-scale integration.