Analog Placement Based on Symmetry-Island Formulation

Po-Hung Lin, Yao-Wen Chang, *Member, IEEE*, and Shyh-Chang Lin

***Abstract*—To reduce the effect of parasitic mismatches and circuit sensitivity to thermal gradients or process variations for analog circuits, some pairs of modules need to be placed symmetrically with respect to a common axis, and the symmetric modules are preferred to be placed at closest proximity for better electrical properties. Most previous works handle the problem with symmetry constraints by imposing symmetric-feasible conditions in floorplan representations and using cost functions to minimize the distance between symmetric modules. Such approaches are inefficient due to the large search space and cannot guarantee the closest proximity of symmetry modules. In this paper, we present *the first* linear-time-packing algorithm for the placement with symmetry constraints using the topological floorplan representations. We first introduce the concept of a symmetry island which is formed by modules of the same symmetry group in a single connected placement. Based on this concept and the B*-tree representation, we propose automatically symmetric-feasible (ASF) B*-trees to directly model the placement of a symmetry island. We then present hierarchical B*-trees (HB*-trees) which can simultaneously optimize the placement with both symmetry islands and nonsymmetric modules. Unlike the previous works, our approach can place the symmetry modules in a symmetry group in close proximity and significantly reduce the search space based on the symmetry-island formulation. In particular, the packing time for an ASF-B*-tree or an HB*-tree is the same as that for a plain B*-tree (only linear) and much faster than previous works. Experimental results show that our approach achieves the best-published quality and runtime efficiency for analog placement.**

***Index Terms*—Analog circuit, floorplanning, physical design, placement.**

I. Introduction

FOR ANALOG layout design, some pairs of modules need to be placed symmetrically with respect to a common axis. The symmetric placement has several advantages: It reduces the effect of parasitic mismatches which may lead to higher offset voltages and degrade power-supply rejection ratio [2]. It can also reduce the circuit sensitivity to process variations by placing the symmetric devices closed to each other. Failure to adequately balance thermal coupling in a differential circuit can even introduce unwanted oscillations [3]. Furthermore, the symmetric modules are preferred to be placed at closest proximity for better parasitic matching and other electrical properties.

A. Previous Work

The problem of analog placement considering symmetry constraints has been extensively studied in the literature. Most of these works used the simulated-annealing (SA) algorithm [4] in combination with floorplan representations to handle symmetry constraints. We can classify these representations into two major categories: 1) the absolute representation and 2) the topological representation.

An absolute representation was proposed by Jepsen and Gellat [5]. For this representation, each module is associated with an absolute coordinate on a gridless plane. It operates on a module by changing its coordinate directly. The KOAN/ANAGRAM II [2], PUPPY-A [6], and LAYLA [7] systems all adopted the absolute representation to handle the placement of analog modules. The main weakness of the absolute method lies in the fact that it may generate an infeasible placement with overlapped modules. Therefore, a postprocessing step must be performed to eliminate this condition, which implies a longer computation time.

Recently, most previous works apply topological floorplan representations due to its flexibility and effectiveness. Balasa *et al.* derived the symmetric-feasible conditions for several popular floorplan representations including sequence pairs (SPs) [8], O-tree [9], and binary trees [10]. To explore the solution space in the symmetric-feasible binary trees, they augmented the B*-tree [11] using various data structures, including segment trees [3], [12], red–black trees [13], and deterministic skip lists [14]. Lin *et al.* [15] also presented the symmetric-feasible conditions for the TCG-S representation. Three more recent works [16]–[18] further took advantage of the symmetric-feasible condition in SPs [8]. Koda *et al.* [16] proposed a linear-programming-based method, and Tam *et al.* [17] introduced a dummy node and additional constraint edges for each symmetry group after obtaining a symmetric-feasible SP. Krishnamoorthy *et al.* [18] proposed an $O(m \cdot n \lg \lg n)$ packing-time algorithm by employing the priority queue, where m is the number of symmetry groups and n is the number of modules. More recently, Zhang *et al.* [19] further improved the perturbation time of the TCG representation from $O(n^2)$ to $O(n)$.

Manuscript received May 31, 2008; revised October 15, 2008 and January 6, 2009. Current version published May 20, 2009. This paper was presented in part at the 2007 ACM/IEEE Design Automation Conference (DAC'07) [1]. This work was supported in part by Springsoft, by Etron, by TSMC, and by the National Science Council of Taiwan under Grants NSC-096-2917-I-002-121, NSC 93-2815-C-002-046-E, NSC 94-2215-E-002-005, and NSC 94-2752-E-002-008-PAE. This paper was recommended by Associate Editor H. E. Graeb.

P.-H. Lin is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: marklin@eda.ee.ntu.edu.tw).

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

S.-C. Lin is with the Physical Design Group, Springsoft, Inc., Hsinchu 300, Taiwan (e-mail: chris_lin@springsoft.com).

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.2009.2017433

Most of the previous works showed that the symmetric-feasible conditions in the topological representations can handle the placement problem with a symmetry group effectively. However, it is time-consuming to generate a relatively larger scale placement with several symmetry groups. For example, Koda *et al.* [16] reported that it takes almost an hour on a 3.2-GHz Pentium PC to generate a symmetric placement of 110 modules with five symmetry groups.

We observed several problems/deficiencies in the previous works for analog placement: First, most previous works employed either an initial scan or a postprocessing with penalty to avoid or fix the violation of the symmetric-feasible condition for each perturbation during the SA process. There is no direct representation that can guarantee the symmetric-feasible condition in the representation itself. It is clear that such approaches are inefficient due to the large search space and fixing overheads. Consequently, the previous works using topological floorplan representations need $O(m \cdot n \lg \lg n)$ time for packing n modules. Second, most previous works used cost functions or weights to penalize the solution with symmetric modules far from each other. Obviously, such approaches cannot guarantee the close proximity (or even the adjacency) of symmetry modules.

B. Our Contributions

In this paper, we present *the first* linear-time-packing algorithm for the placement with symmetry constraints, compared to the previous works that need $O(m \cdot n \lg \lg n)$ time. Specifically, the packing complexity of the previous works [8]–[10], [15]–[17], [19] are all $O(n^2)$ time while those of [3], [12]–[14] need $O(n \lg n)$ time (the work in [18] requires $O(m \cdot n \lg \lg n)$ time, which was the fastest previous work).

We first introduce the concept of *symmetry island* that keeps modules of the same symmetry group connected to each other so that the circuit sensitivity to thermal gradients or process variations can be reduced. Based on this concept and the B*-tree representation [11], we propose a representation called automatically symmetric-feasible (ASF)-B*-trees that can model the compacted placement of a symmetry island (i.e., the symmetric placement of the modules in a symmetry group). Specifically, an ASF-B*-tree corresponds to a symmetry island with a rectilinear placement. It guarantees a symmetric placement and does not need to verify/fix the symmetric-feasible conditions during SA perturbations.

We then present a hierarchical framework called hierarchical B*-trees (HB*-trees) which can simultaneously optimize the placement with both symmetry islands and nonsymmetric modules and dynamically update the rectilinear shape for the modules in a symmetry island. In particular, the overall time complexity for packing an ASF-B*-tree or an HB*-tree is the same as that for a plain B*-tree (only linear) and much faster than previous works. Experimental results based on the MCNC benchmarks [15] and the real industry designs used in [16] show that our approach produces the best published results and runtime efficiency for analog placement. Furthermore, the scalability of our approach is much better than those of the previous works. It should be noted that our formulation requires modules in a symmetry island to be connected, which corresponds to common cases in analog placements. We can leverage this property to prune the solution subspace formed with nonsymmetry-island placements, leading to efficient and effective operations of the ASF-B*-trees. According to our empirical results, in particular, this solution pruning does not degrade the resulting solution quality for practical applications.

TABLE I
COMPARISONS OF POPULAR PREVIOUS WORKS AND OUR APPROACHES. n: THE NUMBER OF MODULES; m: THE NUMBER OF SYMMETRY PAIRS

Analog Placement Approach	Perturbation Time	Packing Time
O-tree [9]	$O(\lg n)$	$O(n^2)$
B*-tree [10]	$O(\lg n)$	$O(n^2)$
B*-tree + Seg. Tree [3]	$O(\lg n)$	$O(n \lg n)$
BT + RB-tree [13]	$O(\lg n)$	$O(n \lg n)$
BT + Skip List [14]	$O(\lg n)$	$O(n \lg n)$
Sequence-pair (SP) [8]	$O(1)$	$O(n^2)$
SP + LP [16]	$O(1)$	$O(n^2)$
SP w. Dummy [17]	$O(1)$	$O(n^2)$
SP w. Priority Queue [18]	$O(1)$	$O(m \cdot n \lg \lg n)$
TCG-S [15]	$O(n^2)$	$O(n^2)$
TCG [19]	$O(n)$	$O(n^2)$
ASF-B*-tree + HB*-tree	$O(\lg n)$	*O(n)*

Table I compares state-of-the-art previous works using the topological floorplan representations and our approach (ASF-B*-tree + HB*-tree).

The remainder of this paper is organized as follows. Section II gives the preliminaries about the symmetry constraints, symmetry islands, and the B*-tree representation. Section III presents how to model the placement of a symmetry group as a symmetry island using the ASF-B*-tree. Section IV proposes the hierarchical framework, HB*-tree, and Section V presents our placement algorithm. Section VI reports the experimental results, and finally, Section VII concludes this paper.

II. PRELIMINARIES

In this section, we first introduce the symmetry constraints for analog placement, the definitions of symmetry types, and the concept of symmetry islands. Then, we review the B*-tree representation in [11] on which this paper is based.

A. Symmetry Constraints

Symmetry constraints can be formulated in terms of *symmetry types*, *symmetry groups*, *symmetry pairs*, and *self-symmetric modules*. In analog layout design, a symmetry group may contain some symmetry pairs and self-symmetric modules with respect to a certain symmetry type. A symmetry type may correspond to a symmetry axis in either horizontal or vertical direction. Fig. 1 shows two different symmetry types with either vertical or horizontal symmetry axis.

For the symmetric placement with the vertical (horizontal) symmetry axis shown in Fig. 1(a) [Fig. 1(b)], a symmetry pair with two modules of the same dimensions and orientations should be placed symmetrically along the vertical (horizontal) symmetry axis. A self-symmetric module whose internal

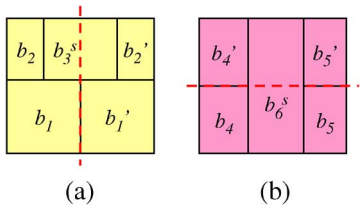


Fig. 1. Two symmetry types. (a) Symmetric placement with the vertical symmetry axis. (b) Symmetric placement with the horizontal symmetry axis.

TABLE II
NOTATIONS IN THIS PAPER

b	a module
S	a symmetry group
(b, b')	a symmetry pair
b^s	a self-symmetric module
b^r	the representative of a symmetry pair or a self-symmetric module
n	number of modules
m	number of symmetry groups
(x_i, y_i)	the center coordinate of the module b_i
w_i, h_i	the width and the height of the module b_i
$\hat{x}_i, \hat{y}_i$	the coordinate(s) of the symmetry axis (axes) of the symmetry group S_i

structure is self-symmetric must have its center placed at the symmetry axis.

We use the notations listed in Table II throughout this paper. Let $S = \{S_1, S_2, \ldots, S_m\}$ be a set of m symmetry groups whose coordinate(s) of the symmetry axis (axes) is (are) denoted by $\hat{x}_i$ or $\hat{y}_i$ ($\hat{x}_i$ and $\hat{y}_i$), $1 \le i \le n$. A symmetry group $S_i = \{(b_1, b_1'), (b_2, b_2'), \ldots, (b_p, b_p'), b_1^s, b_2^s, \ldots, b_q^s\}$ consists of p symmetry pairs and q self-symmetric modules, where (b_j, b_j') denotes a symmetry pair and b_k^s denotes a self-symmetric module. Let (x_j, y_j) and (x_j', y_j') denote the respective coordinates of the centers of two modules b_j and b_j' in a symmetry pair (b_j, b_j'), respectively, and (x_k^s, y_k^s) denotes the coordinate of the center of the self-symmetric module b_k^s. The symmetric placement of a symmetry group S_i with the vertical (horizontal) symmetry axis must satisfy (1) [(2)]

$$\begin{aligned} x_j + x_j' &= 2 \times \hat{x}_i \quad && \forall j = 1, 2, \ldots, p \\ y_j &= y_j' && \forall j = 1, 2, \ldots, p \\ x^s{}_k &= \hat{x}_i && \forall k = 1, 2, \ldots, q \end{aligned} \tag{1}$$

$$\begin{aligned} x_j &= x_j' && \forall j = 1, 2, \ldots, p \\ y_j + y_j' &= 2 \times \hat{y}_i && \forall j = 1, 2, \ldots, p \\ y^s{}_k &= \hat{y}_i && \forall k = 1, 2, \ldots, q. \end{aligned} \tag{2}$$

B. Symmetry Island

Before introducing the symmetry island, we shall first investigate the effect of the symmetric device layout on the electrical matching properties of the symmetric devices. Pelgrom *et al.* [20] measured the mismatch between MOS transistors with various electrical parameters as a function of device areas, distances, and orientations. According to Pelgrom *et al.* [20], the difference of an electrical parameter P between two rectangular devices is modeled by the standard deviation, as shown in (3), where A_P is the area proportionality constant for P, W and L denote the respective width and length of the device, and S_P denotes the variation of P under the device spacing D_x

$$\sigma^2(\Delta P) = \frac{A_P^2}{WL} + S_P^2 D_x^2. \tag{3}$$

We assume that the device dimensions of modules in a symmetry pair are the same. According to the above equation, the larger the distance between the symmetry pair, the greater differences between their electrical properties. Therefore, it is of significant importance for the symmetric devices of a symmetry group to be placed in close proximity. Fig. 2(a) shows an analog circuit of a two-stage CMOS operational amplifier containing the differential input subcircuit. The devices $M1$, $M2$, $M3$, $M4$, and $M5$ in the differential input subcircuit form a symmetry group $S = \{(M1, M2), (M3, M4), M5\}$. Fig. 2(b) and (c) shows two corresponding layouts with different placement styles for the symmetry group S. The layout style in Fig. 2(c) is generally considered much better than that in Fig. 2(b) because the symmetric modules of the same symmetry group are placed at closer proximity (or even adjacent) to each other. Consequently, the sensitivities due to process variations can be minimized, and the circuit performance can be improved.

Based on the placement with the closest proximity for a symmetry group as shown in Fig. 2(c), we introduce the concept of symmetry islands and give its definition as follows.

Definition 1: A symmetry island is a placement of a symmetry group in which each module in the group abuts at least one of the other modules in the same group, and all modules in the symmetry group form a connected placement.

We further use the example in Fig. 3 to explain the concept of symmetry islands. The symmetry group S_1 in Fig. 3(a) forms a symmetry island but that in Fig. 3(b) does not, since it results in two disconnected components. The placement style in Fig. 3(a) is preferred in analog layout design due to its better electrical properties.

C. Review of B*-Trees

Since this paper is based on the B*-tree representation [11], we shall first give a brief review over the representation. A B*-tree is an ordered binary tree representing a *compacted placement*, in which every module can no longer move left and bottom. As shown in Fig. 4, every node of a B*-tree corresponds to a module of a compacted placement. The root of a B*-tree corresponds to the module on the bottom-left corner. For each node n corresponding to a module b, the left child of n represents the lowest adjacent module on the right side of b, while the right child of n represents the first module above b with the same horizontal coordinate.

Given a B*-tree, we can calculate the coordinate of each module by a preorder tree traversal. Suppose the module b_i, represented by the node n_i, has the bottom-left coordinate (x_i, y_i), the width w_i, and the height h_i. Then, for the left child n_j of n_i, $x_j = x_i + w_i$; for the right child n_k of n_i, $x_k = x_i$. In addition, we maintain a contour structure to calculate the y-coordinates. Thus, starting from the root node, whose bottom-left coordinate is (0, 0), then visiting the root's left subtree and,

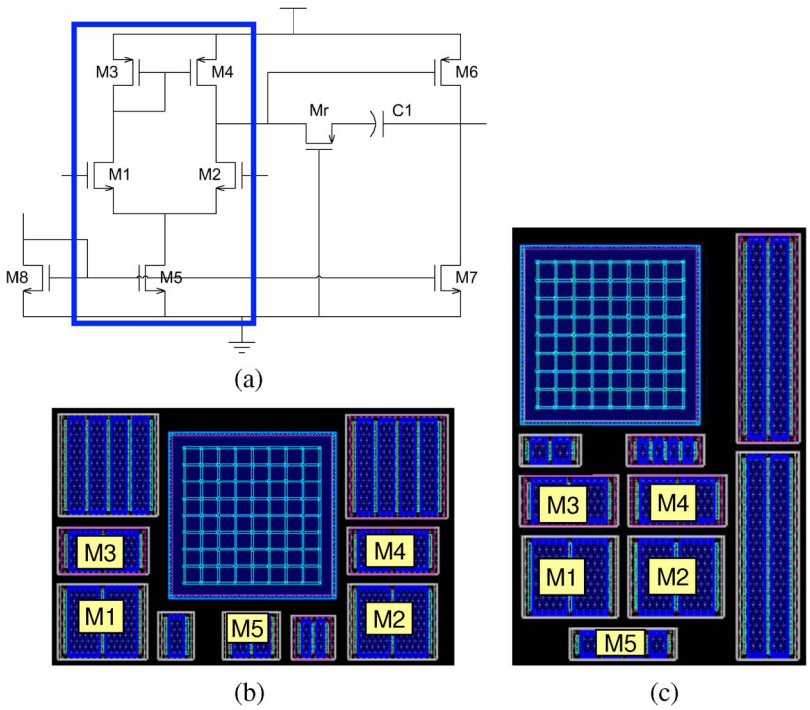


Fig. 2. Example analog circuit and two different layout styles for the circuit. (a) Schematic of a two-stage CMOS operational amplifier, where the differential input subcircuit forms a symmetry group. (b) Layout design of the circuit in (a), where the devices of a symmetry group are not placed close to each other. (c) Another layout design of the circuit in (a), where the devices of a symmetry group are placed close to each other.

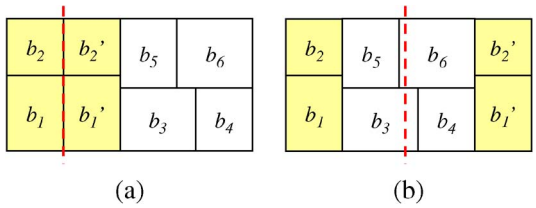


Fig. 3. Two symmetric-placement examples of a symmetry group $S_1 = \{(b_1, b_1'), (b_2, b_2')\}$. (a) S_1 forms a symmetry island. (b) S_1 cannot form a symmetry island.

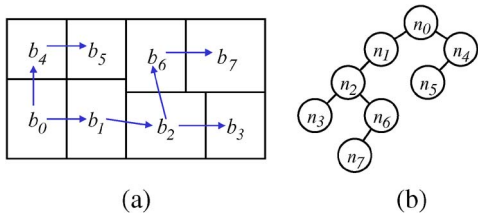


Fig. 4. (a) Compacted placement [same as in Fig. 3(a)]. (b) B*-tree representing the compacted placement in (a).

then, its right subtree, this preorder-tree-traversal procedure, also known as B*-tree packing, calculates all coordinates of the modules in the placement. Using a doubly linked list to implement the contour structure, the total packing time is linear to the number of modules.

III. Placement of a Symmetry Group

In this section, we propose the ASF-B*-tree to consider the symmetric placement of a symmetry group and the packing of the symmetry modules to make a symmetry island. Like

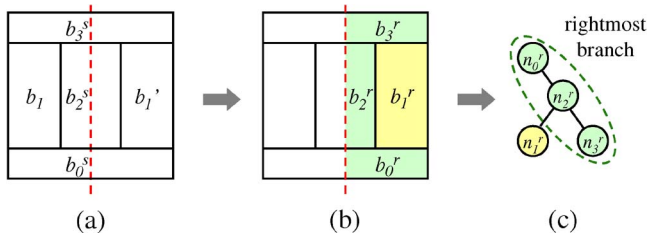


Fig. 5. (a) Placement example of a symmetry group with a vertical symmetry axis. (b) Selecting a representative for each symmetry pair and self-symmetric module. (c) ASF-B*-tree (also a representative B*-tree) representing the placement of the symmetry group, where the dash circled nodes represent the left-boundary modules.

B*-trees, the ASF-B*-tree can represent only *compacted* symmetric placement; in particular, there exists a unique correspondence between a compacted symmetric placement of a symmetry group and its induced ASF-B*-tree which results in a symmetry island. We first present the definitions and properties of the ASF−B*-tree and then prove the correspondence between a symmetry island and its induced ASF-B*-tree.

Before introducing the ASF-B*-tree, we should define the *representative* of a symmetry pair, the representative of a self-symmetric module, and the *representative* B*-*tree*.

Definition 2: The representative b_j^r of a symmetry pair (b_j, b_j') is b_j'.

Definition 3: The representative b_k^r of a self-symmetric module b_k^s is the right (top) half of b_k^s in a symmetric placement with respect to a (horizontal) symmetry axis.

For the example shown in Fig. 5, the representative b_1^r of the symmetry pair $\{b_1, b_1'\}$ is b_1', while the representative b_0^r of the self-symmetric module b_0^s is the right half of b_0^s.

It should be noted that each symmetry pair or self-symmetric module must have its own representative module. Therefore, the number of the representatives in a symmetry group should be the same as the number of symmetry pairs and self-symmetric modules. We define the representative B*-tree as follows.

Definition 4: A representative B*-tree is a B*-tree containing only the representative nodes that correspond to representative modules.

In the following, we describe how to obtain an ASF-B*-tree by making a representative B*-tree symmetric feasible for symmetric placements with vertical and horizontal symmetry axes. We first introduce the *mirrored placement* of the representative modules for a symmetry group.

Definition 5: The mirrored placement of the representative modules for a symmetry group S_i is to place the nonrepresentative modules on the mirrored positions of the representative ones for each symmetry pair or each self-symmetric module in S_i with respect to its symmetry axis (axes). Furthermore, the representative and the nonrepresentative modules of each self-symmetric module are not disjointed.

We are now ready to define the symmetric-feasible condition of a representative B*-tree for the symmetric placements.

Definition 6: A representative B*-tree is symmetric feasible if the mirrored placement of the representative modules can be obtained after packing the representative B*-tree.

In Fig. 5(a), the modules in the symmetry group $S = \{(b_1, b_1'), b_0^s, b_2^s, b_3^s\}$ are placed symmetrically with respect to the vertical axis. To construct the corresponding representative B*-tree, we should select the representative module of each symmetry pair and self-symmetric module and consider the placement on the right half-plane. Fig. 5(b) shows the representative modules, and Fig. 5(c) shows the corresponding representative B*-tree of the symmetric placement. Each node in the representative B*-tree corresponds to a representative module.

To make the representative B*-tree symmetry symmetric feasible, we have the following lemmas which gives the symmetry condition for a self-symmetric module and a symmetry pair.

Lemma 1: The representative of a self-symmetric module must abut the symmetry axis.

Proof: Let S be a symmetry group with a vertical symmetry axis and b^s be a self-symmetric module in S. The symmetry axis of S is denoted by $\hat{x}$, and the center of b^s is denoted by (x^s, y^s).

Based on (1), the symmetry axis $\hat{x}$ always passes through the center (x^s, y^s) of the self-symmetric module b^s, i.e., $\hat{x} = x^s$. According to Definition 3, the representative b^r of b^s is the right half of b^s. Therefore, the center (x^s, y^s) of b^s must be on the left boundary of b^r. To keep the symmetric-feasible condition $\hat{x} = x^s$, b^r must abut the symmetry axis $\hat{x}$. The case for a symmetry group with a horizontal symmetry axis can be proved similarly. Q.E.D.

Lemma 2: The representative of a symmetry pair not on a symmetry axis is always symmetric feasible.

Proof: Let S be a symmetry group with a vertical symmetry axis and (b, b') be a symmetry pair in S. The symmetry axis of S is denoted by $\hat{x}$. The respective centers of b and b' are (x, y) and (x', y'), and the respective widths/heights of b and b' are w/h and w'/h', where $w = w'$ and $h = h'$. The representative of the symmetry pair (b, b') is b'.

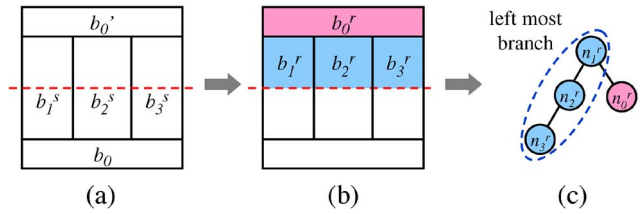


Fig. 6. (a) Placement example of a symmetry group with a horizontal symmetry axis. (b) Selecting a representative module for each symmetry pair and self-symmetric module. (c) ASF-B*-tree (also a representative B*-tree) representing the placement of the symmetry group, where the dash circled nodes represent the bottom-boundary modules.

Given the coordinate of the representative b' and the vertical symmetry axis $\hat{x}$, the coordinate of the symmetric module b can be calculated by (1). We have $x = 2 \times \hat{x} - x'$ and $y = y'$. After transposing $\hat{x}$ to the left side and having the absolute value on both sides, we have $|x - \hat{x}| = |\hat{x} - x'|$. Since the representative is not on the symmetry axis, we have $|x - \hat{x}| = |\hat{x} - x'| \geq w/2$. It means that the distances from the symmetry axis to the centers of b and b' are greater than or equal to half of the width of b or b'. Since b and b' are on different sides of the symmetry axis, b and b' will not overlap each other. Therefore, the symmetric-feasible condition is always satisfied. The case for a symmetry group with a horizontal symmetry axis can be proved similarly. Q.E.D.

According to Lemma 1 and the boundary constraints [21] in the B*-trees, we have the following property for the symmetric-feasible representative B*-trees representing 1-D symmetric placement.

Property 1: The left-boundary (right-boundary) constraint for the symmetric placement with respect to a vertical (horizontal) symmetry axis: The representative node of a self-symmetric module should always be on the rightmost (leftmost) branch of the representative B*-tree.

Based on the above property, the nodes representing the modules on the left boundary should be on the rightmost branch, as shown in Fig. 5(c).

Similarly, we can get the symmetric-feasible representative B*-tree of the symmetric placement when the symmetry axis is in the horizontal direction. In this case, we only consider the top half-plane during the placement of the representative modules. Fig. 6(c) shows the representative B*-tree of the symmetry group $S = \{(b_0, b_0'), b_1^s, b_2^s, b_3^s\}$ having the symmetric placement with respect to the horizontal symmetry axis in Fig. 6(a). Again, the representatives of the self-symmetric modules should abut the horizontal symmetry axis which is on the bottom boundary of the top half-plane. Therefore, the nodes representing the modules on the bottom boundary should be on the leftmost branch, as shown in Fig. 6(c).

Based on Definition 4 and Property 1, we define an ASF-B*-tree as follows.

Definition 7: An ASF-B*-tree is a representative B*-tree which satisfies Property 1.

Once an ASF-B*-tree is packed, the coordinates of these representatives are obtained. We can further calculate the

coordinates of their symmetric modules based on (1) and (2) with the given coordinates of the symmetry axes, $\hat{x}_i$ and $\hat{y}_i$. Then, we have the symmetric placement of a symmetry group, and it automatically forms a symmetry island.

Based on Lemmas 1 and 2, we have the following theorems.

Theorem 1: An ASF-B*-tree is symmetric feasible in a symmetric placement of a symmetry group with respect to either a vertical or a horizontal symmetry axis.

Proof: An ASF-B*-tree is symmetric feasible if all the representatives in the ASF-B*-tree are symmetric feasible. There are four kinds of representatives, and the symmetric-feasible condition for each is defined and proved in Lemmas 1 and 2. Therefore, an ASF-B*-tree is symmetric feasible in a symmetric placement of a symmetry group with respect to either a vertical or a horizontal symmetry axis. Q.E.D.

Theorem 2: The packing of an ASF-B*-tree results in a symmetry island of the corresponding symmetry group.

Proof: It is obvious that all the representative modules will form a connected placement after packing. We set the coordinate(s) of the symmetry axis (axes) to the left or (and) the bottom boundary (boundaries) of the connected placement of the representative modules. The coordinates of the symmetric modules can be calculated by (1) and (2). The symmetric modules also form a connected placement, and the boundary of the connected placement also abut the symmetry axis (axes). Therefore, the whole symmetry group forms a connected placement, and each module in the group abuts at least one of the other modules in the same group. The packing of an ASF-B*-tree, thus, results in a symmetry island of the corresponding symmetry group. Q.E.D.

Theorem 3: There exists a unique correspondence between a compacted symmetric placement of a symmetry group and its induced ASF-B*-tree.

Proof: According to Chang *et al.* [11], there is a unique correspondence between an admissible placement and its induced B*-tree. After obtaining the placement of the representative modules, we simply get the mirrored placement of the symmetric ones. The mirrored placement is also unique. Therefore, there exists a unique correspondence between a compacted symmetric placement of a symmetry group and its induced ASF-B*-tree. Q.E.D.

Based on the earlier theorems, we can correctly find a corresponding symmetric placement for an ASF-B*-tree very efficiently, by avoiding searching in redundant solution spaces. It will be clear later in Section VI that these nice properties of ASF-B*-trees lead to superior solution quality and efficiency for analog placement.

IV. Hierarchical Framework

We propose a hierarchical framework, called *hierarchical* B**-tree* (HB**-tree* for short), to handle the simultaneous placement of modules in symmetry islands and nonsymmetric modules. In an HB*-tree, the symmetry island of each symmetry group can be in any rectilinear shapes, and symmetry and nonsymmetric modules are simultaneously placed to optimize the placement.

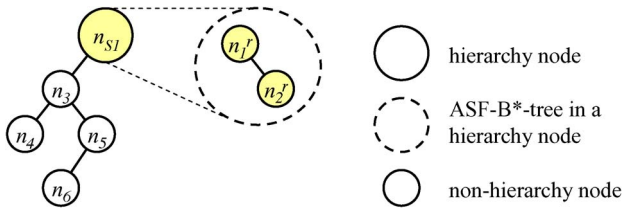


Fig. 7. HB*-tree for the placement in Fig. 3(a).

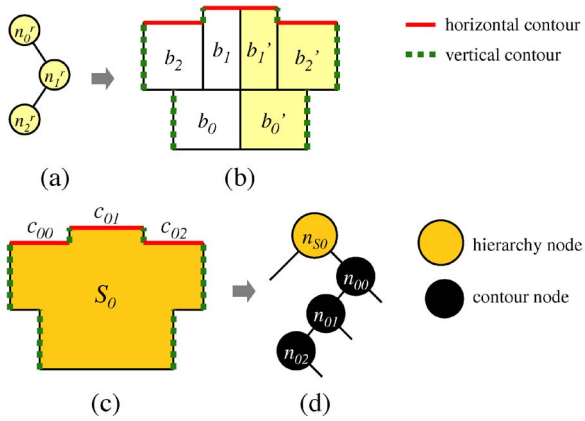


Fig. 8. (a) ASF-B*-tree of a symmetry group S_0. (b) Horizontal and vertical contours of the corresponding placement. (c) Symmetry island and its effective contours. (d) HB*-tree for the rectilinear symmetry island.

A. HB*-Tree Representation

Fig. 7 shows an HB*-tree for the placement in Fig. 3(a). Two symmetry groups, S_1 and S_2, are represented by two hierarchy nodes, n_{S_1} and n_{S_2}, and each hierarchy node contains an ASF-B*-tree that corresponds to a symmetry island in the symmetric placement.

The symmetry islands are often not rectangular but are of rectilinear shapes. For example, in Fig. 8(c), the symmetry island of the symmetry group S_0 is of the rectilinear shape. Therefore, we should augment the HB*-tree in Fig. 7 to handle rectilinear symmetry islands. Wu *et al.* [22] proposed a method to deal with rectilinear modules by slicing a rectilinear module into several rectangular submodules along each vertical boundary. However, it is complicated to maintain the relationship between the submodules during B*-tree perturbations.

Instead of slicing a rectilinear symmetry island, we introduce *contour nodes* to represent top horizontal contour segments of the symmetry island. In Fig. 8(c), there are three horizontal contour segments: c_{00}, c_{01}, and c_{02}. We augment the HB*-tree by introducing the three contour nodes, n_{00}, n_{01}, n_{02}, as shown in Fig. 8(d). Each contour node keeps the coordinates of the corresponding horizontal contour segment. The relationship of a hierarchy node, its contour nodes, and other regular module nodes is described as follows.

Property 2: Properties for an HB*-tree.

1) The left child of a hierarchy node, if any, must be a noncontour node.
2) The right child of a hierarchy node must be the contour node representing the leftmost top horizontal contour segment of the symmetry island.

3) The left child of a contour node, if any, must be the contour node representing the next contour segment on the right side.
4) The right child of a contour node, if any, must be a noncontour node.
5) The children of a regular module node must be noncontour nodes.
6) The parent of a contour node cannot be a regular module node.

Proof: Given a symmetry group S_0, b_{S_0} denotes the symmetry island of S_0, n_{S_0} denotes the corresponding hierarchy node, and n_{0i} represents the ith top contour segment of b_{S_0} from left to right.

1) Since the contour node n_{0i} represents the ith top contour segment of b_{S_0}, it is impossible for n_{0i} to be the left child of n_{S_0} that corresponds to the lowest adjacent module on the right side of b_{S_0}, based on the B*-tree definition. The property thus follows.
2) According to the definition of the B*-tree, the right child of n_{S_0} represents the first module above b_{S_0}. Since the top horizontal contour segments of b_{S_0} always abut b_{S_0}, other modules cannot be placed between b_{S_0} and its top contour segments. Therefore, the right child of n_{S_0} must be a contour node representing the leftmost top horizontal contour segment of b_{S_0}.
3) By the contour-node definition, the contour node $n_{0,i}$ represents the ith top contour segment of b_{S_0} from left to right, and the left child of $n_{0,i}$, if any, is $n_{0,i+1}$, representing the next [$(i+1)$th] contour segments. If $n_{0,i}$ represents the last (the rightmost) top contour segment, the left child of n_{0i} is empty.
4) The right child of the contour node n_{0i} represents the first module above the ith top contour segment of b_{S_0}. If there exists another contour node n_{0j} that is the right child of n_{0i}, both contour segments will overlap each other with n_{0j}'s contour segment on top of that of n_{0i}, implying that n_{0i} is not a contour node. A contraction.
5) Based on the second and the third properties of the HB*-tree, the contour node n_{0i} cannot be the left or right child of a regular module node. The property thus follows.
6) Based on the construction of the HB*-tree, the parent of a contour node is either a contour node or a hierarchy node.

Q.E.D.

Fig. 8(a) shows the ASF-B*-tree of the symmetry group $S_0 = \{(b_0, b_0'), (b_1, b_1'), (b_2, b_2')\}$. In Fig. 8(b), the horizontal and vertical contours are obtained from the rectilinear outline after packing the ASF-B*-tree. Fig. 8(c) shows the symmetry island and the effective horizontal and vertical contours. The horizontal contour segments are denoted as c_{00}, c_{01}, and c_{02} from left to right. Therefore, we have a hierarchy node n_{S_0} representing the symmetry island of the symmetry group S_0, and three contour nodes n_{00}, n_{01}, and n_{02} representing the contour segments. The relationship between the hierarchy node and its contour nodes is shown in the HB*-tree in Fig. 8(d).

After introducing the representation and the properties of HB*-trees, we present the packing procedure for ASF-B*-trees and HB*-trees.

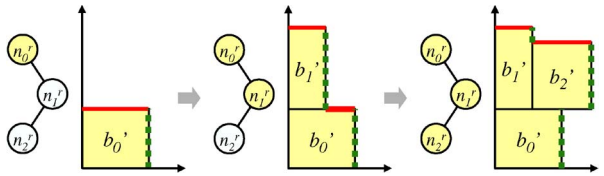


Fig. 9. Packing procedure including the contour updates of the ASF-B*-tree in Fig. 8(a).

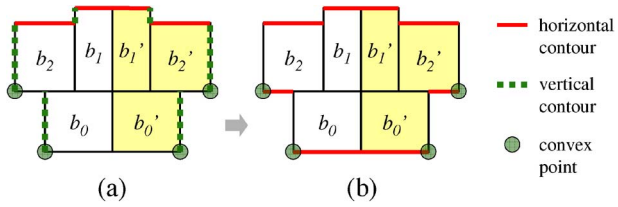


Fig. 10. Generation of the bottom contour of the symmetry island based on the dual vertical contours. (a) Convex points obtained by traversing the dual vertical contours from bottom to top. (b) Bottom horizontal contour connected by the convex points.

B. ASF-B*-Tree Packing

The packing of the ASF-B*-tree is similar to that of the B*-tree [11] which follows the preorder-tree-traversal procedure to calculate the coordinates of the modules. During the packing, two double-linked lists are implemented to keep both horizontal and vertical contour structures. Fig. 9 shows the packing procedure of the example ASF-B*-tree in Fig. 8(a). The bold (red) lines denote the horizontal contour, while the dotted (green) lines represent the vertical contour.

After obtaining the coordinates of all representative modules in the symmetry group, we can calculate the coordinates of the symmetric modules and the extended contours based on either (1) or (2). Fig. 8(b) shows the resulting placement of the symmetry group and the contours of the symmetry island for the ASF-B*-tree shown in Fig. 8(a). As shown in Fig. 8(b), the symmetry island contains one top horizontal and dual vertical contours. To further calculate the bottom horizontal contour of the symmetry island, we need to traverse both vertical contours from bottom to top and keep the convex points as shown in Fig. 10(a). By connecting the convex points horizontally, we can obtain the bottom horizontal contour of the symmetry island, as shown in Fig. 10(b).

C. HB*-Tree Packing

The HB*-tree packing also adopts the preorder-tree-traversal procedure. When a hierarchy node is traversed, the ASF-B*-tree in the hierarchy node should be packed first to obtain the contours of the symmetry island described previously. The contours are then stored in the corresponding hierarchy node. During packing a hierarchy node representing a symmetry island, we should calculate the best packing coordinate for the bottom boundary of the symmetry island, based on the bottom contour shown in Fig. 10(b). We then proceed to pack the left child of the hierarchy node. After the left child and all its descendants are packed, we pack the first contour node of the symmetry island, followed by the second one, and so on. When packing the contour nodes, we only need to update their

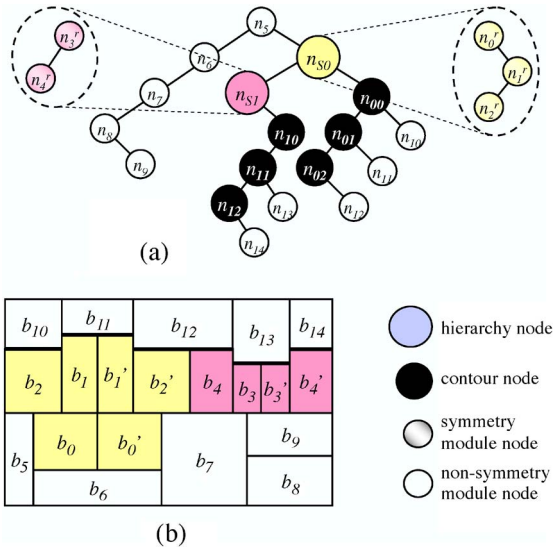


Fig. 11. (a) HB*-tree representing 20 modules with two symmetry groups S_0 and S_1. (b) Resulting placement after packing the HB*-tree.

coordinates and replace the hierarchy node in the contour data structure of the HB*-tree.

Fig. 11(a) shows an HB*-tree representing 20 modules with two symmetry groups S_0 and S_1. For the packing, the two ASF-B*-trees in n_{S_0} and n_{S_1} are packed first, and the rectilinear outlines of the two symmetry islands are obtained. Then, the nodes, n_5, n_6, n_7, n_8, and n_9, are packed in the depth-first search order. The temporal contour list is $\langle n_5, n_6, n_7, n_9 \rangle$. By calculating the rectilinear outlines between the temporal contour list and the bottom boundary of the symmetry island S_0, the dead space between the previously packed modules and the symmetry island can be minimized. The updated temporal contour list becomes $\langle n_{S_0}, n_7, n_9 \rangle$. Continuing the packing procedure, we can obtain the resulting placement of the HB*-tree in Fig. 11(b) finally. Although the purpose of the packing is to obtain a compacted placement, we might need to allocate sufficient white space for the surrounding wells or guard rings based on the device types, such as NMOS or PMOS transistors. When packing a node, the device type of the corresponding module should be compared with those of the previously packed modules in the current contour list. If the device types are different, the currently packed module should be snapped to a position to reserve sufficient white space for the surrounding wells or guard rings.

We have the following theorem for the packing complexity.

Theorem 4: The packing for an ASF-B*-tree or an HB*-tree takes linear time.

Proof: Given a design with n modules (including symmetry and nonsymmetry ones) and m symmetry groups, let $\hat{n}$ be the number of nonsymmetric modules and $n(S_i)$ be the number of modules in each symmetry group S_i, where $n(S_i) \geq 1$. We have $n = \hat{n} + \sum_{i=1}^{m} n(S_i)$.

For the HB*-tree representing the symmetric placement of the given design, there are m hierarchy nodes, $O(\sum_{i=1}^{m} n(S_i))$ contour nodes, and $\hat{n}$ module nodes. For the ASF-B*-tree of the symmetry group S_i in a hierarchy node, there are $O(n(S_i))$ representative nodes.

We first consider the packing for the ASF-B*-tree of the symmetry group S_i in a hierarchy node. It consists of two steps. The first step is the packing for all representative modules. The second step is the calculation of the coordinate of each symmetric module.

According to Chang *et al.* [11], the packing for a B*-tree takes linear time, so the time complexity of the first step is $O(n(S_i))$. Since it takes constant time to calculate the coordinate of a symmetric module, it also takes $O(n(S_i))$ time to compute the coordinates of all the symmetric modules in S_i. Combining both steps, we have the $O(n(S_i))$ time complexity for the packing of an ASF-B*-tree of S_i.

Second, we consider the packing for the HB*-tree. If all the symmetry islands of m symmetry groups are in a rectangular shape. We can ignore the contour nodes in the HB*-tree, and it takes $O(m + \hat{n})$ time to pack the HB*-tree. However, if any symmetry island is in a rectilinear shape, we need to consider the packing of the hierarchy node representing this symmetry island, particularly the additional contour nodes.

The bottom contour of the symmetry island of S_i is obtained when the corresponding ASF-B*-tree of the symmetry group is packed, and the number of the bottom contour segments is $O(n(S_i))$. By comparing the current packing contour segments and the bottom contour segments of the symmetry island from left to right, it also takes $O(n(S_i))$ time to get the coordinates of the modules in the symmetry island S_i.

To sum up, it takes $O(m + \sum_{i=1}^{m} n(S_i) + \hat{n})$ time to pack the HB*-tree. Since $n = \sum_{i=1}^{m} n(S_i) + \hat{n}$, the packing time can be reduced to $O(m + n)$ time. Since the number of symmetry group m is upper bounded by the number of total modules n, the packing time is $O(n)$. Q.E.D.

It should be noted that this is the fastest algorithm in the literature for the placement with symmetry constraints, as shown in Table I.

D. Advanced Symmetry Constraints

For some analog layout applications, the symmetry constraints could be even more complex than what we have considered. We brief the handling of two kinds of such symmetry constraints in the following.

1) Multiple Symmetry-Group Alignment: In some analog layouts, the symmetry axes of different symmetry groups are required to be aligned to share a common symmetry axis. To align multiple symmetry groups with respect to a common vertical (horizontal) symmetry axis, we can insert a zero-height (zero-width) dummy block right at the left (bottom) of each to-be-aligned symmetry island. We then introduce a dummy node as the parent of the hierarchy node representing the corresponding symmetry island in the HB*-tree, where the hierarchy node is the left (right) child of the dummy node. By adjusting the width (height) of each dummy block, the symmetry islands of different symmetry groups can be aligned with respect to a common vertical (horizontal) symmetry axis. Such an alignment technique is an extension of the work in [23].

2) Hierarchical Symmetry: In some fully symmetric analog designs, the device layouts should be hierarchically symmetric.

A symmetry group S_i may also contain a self-symmetry group S_j^s and/or a symmetry-group pair (S_k, S_k'). Consequently, the top-level symmetry group S_{Top} contains all device modules and other symmetry groups hierarchically. Based on the proposed symmetry-island and tree formulations, a hierarchical tree structure [24] that mixes both the ASF-B*-trees and the HB*-trees can be constructed. The optimized fully symmetric placement with the hierarchical symmetry constraint can then be obtained by searching a desired configuration of the tree structure and packing the trees to form the symmetry islands hierarchically.

E. Application to Hierarchical Clustering Constraint

Besides handling the symmetry constraints based on the symmetry-island formulation, the proposed hierarchical framework, HB*-trees, can also effectively manage the hierarchical clustering constraint in analog placement or mixed-signal floorplanning based on the intrinsic hierarchical tree structure.

Let $C = \{C_1, C_2, \ldots, C_l\}$ be a set of device module clusters. Each cluster contains at least two modules, or one module and one of the other clusters, or two of the other clusters. If the cluster C_i contains the cluster C_j, we call C_i a supercluster and C_j a subcluster. The hierarchical clustering constraint limits all the device modules and/or subclusters of the same supercluster to a connected placement.

To formulate the hierarchical clustering constraint using the HB*-trees, each of the hierarchy nodes $n_{C_1}, n_{C_2}, \ldots, n_{C_l}$ denotes a cluster. Each hierarchy node n_{C_i} further contains another HB*-tree to represent the topological relation of the device modules and/or the subclusters in the supercluster denoted by n_{C_i}. After hierarchically constructing the HB*-trees, the placement can be optimized by searching a desired configuration of the HB*-trees while the inner placement of each cluster is connected.

F. Consideration of Nonsymmetry-Island Placements

In addition to the preferred symmetry-island placements in analog layouts, the proposed ASF-B*-trees and HB*-trees can also generate a nonsymmetry-island placement by integrating nonsymmetric modules as a self-symmetric module cluster or a symmetry pair consisting of two module clusters in a symmetry group represented by an ASF-B*-tree. Fig. 12 shows two examples, including the symmetric placements and the corresponding ASF-B*-trees, which integrate nonsymmetric module clusters into symmetry groups. In Fig. 12(a), the nonsymmetric modules, b_3 and b_4, form the self-symmetric module cluster C_1 in the symmetry group S_1. After packing the B*-tree representing the placement of the nonsymmetric modules, the representative node $n_{C_1}^r$ is introduced in the ASF-B*-tree representing a symmetric placement of S_1. Similarly, in Fig. 12(b), the nonsymmetric modules, b_7, b_8, and b_9, form two clusters, C_2 and C_2', as a symmetry pair in the symmetry group S_2. In the corresponding ASF-B*-tree, the representative node $n_{C_2}^r$ is introduced to denote the larger dimensions of the placements of C_2 and C_2'.

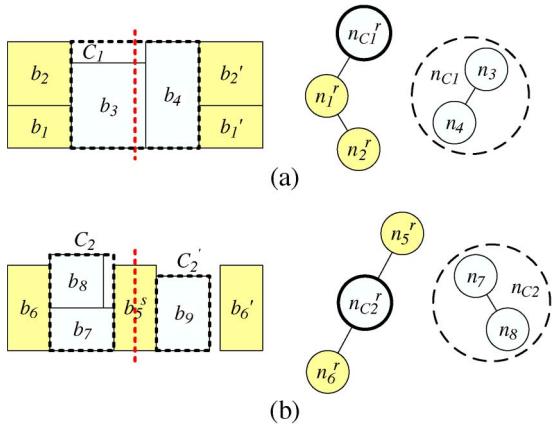


Fig. 12. Integrating nonsymmetric modules into symmetry groups. (a) Nonsymmetric modules form the self-symmetric module cluster $C_1 = \{b_3, b_4\}$ in the symmetry group $S_1 = \{(b_1, b_1'), (b_2, b_2'), C_1^s\}$. (b) Nonsymmetric modules form two clusters, $C_2 = \{b_7, b_8\}$ and $C_2' = \{b_9\}$, as a symmetry pair in the symmetry group $S_2 = \{b_5^s, (b_6, b_6'), (C_2, C_2')\}$.

V. Algorithm

Our algorithm is based on the SA [4]. Given a set of modules and symmetry constraints as the inputs, we construct an initial solution represented by an HB*-tree and, then, perturb it to search for a desired configuration until a predefined termination condition is satisfied. The cost function $\Phi(P)$ of the placement is defined in (4), where α and β are user-specified parameters, A_P is the area of the bounding rectangle for the placement, and W_P is the half-perimeter wire length

$$\Phi(P) = \alpha \times A_P + \beta \times W_P. \tag{4}$$

A. HB*-Tree Perturbation

We apply the following operations to perturb an HB*-tree.

1) Op1: Rotate a module.
2) Op2: Move a node to another place.
3) Op3: Swap two nodes.

In the perturbation, the nonhierarchy nodes have higher probabilities to be selected because rotating, moving, or swapping the hierarchy nodes might incur a big jump in finding the next solution. It is well known that such a big jump might deteriorate the solution quality during the SA process. It should be noted that, due to the special structure of the HB*-tree, we cannot move a nonhierarchy node to the right child of a hierarchy node or the left child of a contour node. The contour nodes are always moved along with its hierarchy node which cannot be moved individually.

B. ASF-B*-Tree Perturbation

In addition to the aforementioned Op1, Op2, and Op3 for HB*-tree perturbation, we introduce the operations, Op4 and Op5, to perturb the ASF-B*-trees. It should be noted that Property 1 should always be satisfied when perturbing an ASF-B*-tree according to the definition of the ASF-B*-trees in Definition 7.

1) Op4: Change a representative.
2) Op5: Convert a symmetry type.

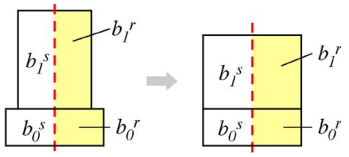


Fig. 13. Rotating the self-symmetric module b_1^s in the symmetry group $S = \{b_0^s, b_1^s\}$ results in the shape change of its representative b_1^r.

1) Module Rotation: When rotating modules in a symmetry group, the corresponding ASF-B*-tree is unchanged. We should consider two cases of symmetry-module rotation.

Case 1) Rotate a symmetry pair.

Case 2) Rotate a self-symmetric module.

In case 1), both modules of a symmetry pair should be rotated at the same time so that they can still be symmetrically placed with respect to a symmetry axis. In case 2), after rotating a self-symmetric module, the shape of its representative should be updated accordingly, as shown in Fig. 13.

2) Node Movement: When moving a node to another place in an ASF-B*-tree, we should consider the following two cases.

Case 1) Move a node representing the representative of a symmetry pair.

Case 2) Move a node representing the representative of a self-symmetric module.

In case 1), we can move the representative node of a symmetry pair to anywhere in an ASF-B*-tree. In case 2), however, we can only move the representative node of a self-symmetric module along the rightmost (leftmost) branch of the ASF-B*-tree for vertical (horizontal) symmetric placement so that Property 1 is satisfied.

3) Node Swapping: When swapping two nodes in an ASF-B*-tree, we consider the following two cases.

Case 1) Both nodes represent the representatives of two different symmetry pairs.

Case 2) At least one node represents the representative of a self-symmetric module.

In case 1), we can arbitrarily swap two nodes representing the representatives of two different symmetry pairs. However, we should be very careful for case 2). If at least one of the swapped nodes represents the representative of a self-symmetric module, the other node must be located on the same branch (i.e., the leftmost or the rightmost branch) of the ASF-B*-tree. Therefore, Property 1 is still satisfied after node swapping.

4) Representative Change: The purpose of changing a representative for a symmetry pair or a self-symmetric module is to optimize the wire length, while the area is kept unchanged after changing the representative. We can change the representative of either a symmetry pair or a self-symmetric module.

Case 1) Change the representative of a symmetry pair.

Case 2) Change the representative of a self-symmetric module.

In case 1), for a symmetry pair (b_j, b'_j), we can simply change the representative from b_j to b'_j or from b'_j to b_j. Fig. 14 shows that changing the representative of the symmetry pair (b_1, b'_1) from b'_1 to b_1 may result in shorter wire length between b_1 and b_3. Similarly, in case 2), for a self-symmetric module b_k^s, we can change its representative by flipping it horizontally or vertically according to its symmetry axis. As shown in Fig. 15, changing the representative of the self-symmetric module b_1^s by flipping it horizontally may result in shorter wire length between b_1^s and b_3. Obviously, each operation takes constant time.

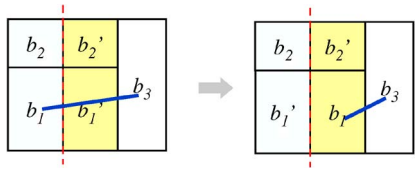


Fig. 14. Changing the representative of the symmetry pair (b_1, b'_1) from b'_1 to b_1 may result in shorter wire length between b_1 and b_3.

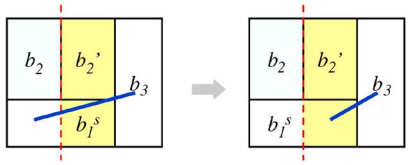


Fig. 15. Changing the representative of the self-symmetric module b_1^s may result in shorter wire length between b_1^s and b_3.

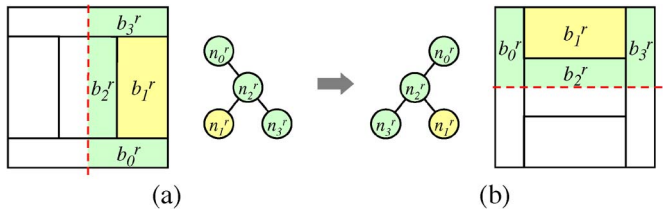


Fig. 16. Converting the symmetry type from (a) vertical symmetry to (b) horizontal symmetry.

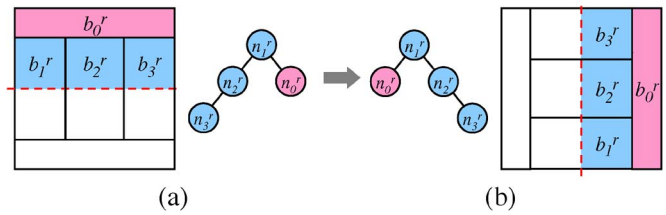


Fig. 17. Converting the symmetry type from (a) horizontal symmetry to (b) vertical symmetry.

5) Symmetry-Type Conversion: For symmetry-type conversion of a symmetry group, we should consider both conversions between the vertical symmetry and the horizontal one.

Case 1) Convert the symmetry type from vertical symmetry to horizontal one.

Case 2) Convert the symmetry type from horizontal symmetry to vertical one.

To convert the symmetry type of a symmetry group from vertical symmetry to horizontal one or vice versa, we first rotate every module including the representative and, then, swap the left and the right children of each node in the given ASF-B*-tree. Figs. 16 and 17 show the respective examples for the conversions of cases 1) and 2).

It should be noted that the symmetry type is usually predefined based on the power/ground lines or signal flows in the layout by the analog designers. Therefore, Op5 is seldom applied in real applications.

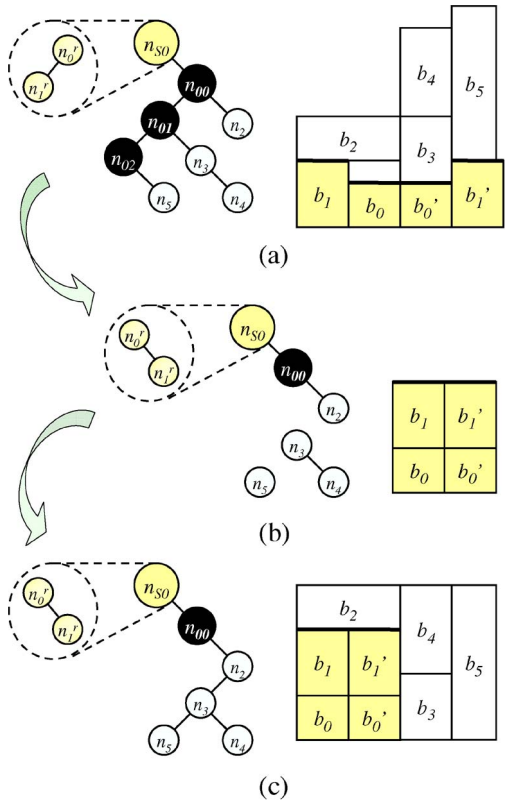


Fig. 18. Example of updating contour-related nodes. (a) HB*-tree and its corresponding placement containing the symmetry group $S_0 = \{(b_0, b_0'), (b_1, b_1')\}$. (b) Intermediate HB*-tree after perturbing the ASFB*-tree in the hierarchy node n_{S0} and the corresponding symmetry island of S_0. The contour-related nodes, n_3 and n_5, become dangling. (c) HB*-tree after updating the contour-related nodes and its corresponding placement.

C. Contour-Node-Related Updates

Once an ASF-B*-tree is perturbed, the number of the corresponding contour nodes in the HB*-tree might be changed. The tree structure might have to be updated accordingly. If the number of contour nodes representing the horizontal contour segments of the symmetry island is increased, the structure of the HB*-tree can be kept unchanged. However, if that of the contour nodes is decreased, some other nodes in the HB*-tree might not have parents. We call such nodes *dangling node*, and we should reassign new parents for these nodes. To keep the relative placement topology before and after perturbing an ASF-B*-tree, we first find the nearest contour node for each dangling node. If the nearest contour node has no right child, it is the parent of the dangling node, and the dangling node will be its right child. If the nearest contour node has a right child, we continuously traverse the leftmost skewed child of the right child. The leftmost skewed child will be the parent of the dangling node, and the dangling node is assigned to its left child. It takes amortized constant time to update the contour-related nodes.

Fig. 18 shows an example of updating contour-related nodes. In Fig. 18(a), there are initially three contour nodes representing the three top contour segments of the symmetry island of the symmetry group S_0. After performing Op2 to perturb the ASF-B*-tree in n_{S_0}, the representative node n_1^r is moved from the left child to the right child of the other representative node n_0^r.

TABLE III
MCNC BENCHMARK CIRCUITS

Circuit	# of Mod.	# of Sym. Mod.	Mod. Area (mm^2)
apte	9	8	46.56
hp	11	8	8.83
ami33	33	6	1.16
ami49	49	4	35.45

TABLE IV
INDUSTRY BENCHMARK CIRCUITS

Circuit	# of Mod.	# of Sym. Mod.	Mod. Area ($10^3 \mu m^2$)
biasynth_2p4g	65	8+12+5	4.70
lnamixbias_2p4g	110	16+6+6+12+4	46.00

The placement of S_0 forms a new symmetry island, as shown in Fig. 18(b) which has only one top contour segment. Therefore, the contour nodes n_{01} and n_{02} disappear, and the nodes n_3 and n_5 become dangling nodes. We first find the nearest contour node of n_3, which is n_{00}. Since n_{00} already has the right child n_2, the leftmost skewed child of n_2 should be searched. In this case, we directly assign n_3 to be the left child of n_2 because n_2 has no left child. After n_3 is assigned to a proper tree location, the nearest contour node of n_5 is then searched, which is also n_{00}. Since n_{00} already has the right child n_2, the leftmost skewed child is searched, which is n_3. We assign n_3 to be the parent of n_5, and n_5 is the left child of n_3.

VI. EXPERIMENTAL RESULTS

We implemented our placement algorithm in the C++ programming language on a 3.2-GHz Intel Pentium4 PC under the Linux operating system. We performed two sets of experiments: One is based on the four MCNC benchmarks (apte, hp, ami33, and ami49) used in [15], and the other consists of two real industry analog designs (biasynth_2p4g and lnamixbias_2p4g) used in [12] and [16] (note that they both were extracted by Koda *et al.* [16] from [12, Figs. 9 and 10]). Table III lists the names of the MCNC benchmark circuits ("Circuit"), the numbers of modules ("# of Mod."), the numbers of symmetry modules ("# of Sym. Mod."), and the total module areas ("Mod. Area"). Table IV lists the names of the industry benchmark circuits ("Circuit"), the numbers of modules ("# of Mod."), the numbers of symmetry modules ("# of Sym. Mod."), and the total module areas ("Mod. Area").

Our approach is based on SA. A left skewed HB*-tree was constructed as the initial solution. The initial temperature T_0 was calculated by (5), where Δ_{avg} is the average uphill cost and P is the initial probability to accept uphill solutions. During the SA process, the temperature was reduced at the rate of 0.9 for each subsequent pass, and 20 000 iterations were performed at each temperature/pass

$$T_0 = -\Delta_{\text{avg}} / \ln P. \tag{5}$$

In the first set of experiments, we compared our algorithm with the following works: SPs [8], segment trees [3], TCG-S [15], and SPs with dummy nodes [17]. Table V lists the names of the MCNC benchmark circuits ("Circuit"), the total areas

TABLE V
COMPARISONS OF AREA UTILIZATION AND CPU TIMES FOR SP (ON SUN SPARC ULTRA-60 433 MHz), SEGMENT TREE (SEG. TREE) (ON SUN SPARC ULTRA-60 433 MHz), TCG-S (ON SUN SPARC ULTRA-60 433 MHz), SP WITH DUMMY NODES (SP w. DUMMY) (ON PENTIUM4 3.2 GHz), AND OUR HB*-TREE (ON PENTIUM4 3.2 GHz) WITH AREA OPTIMIZATION ALONE, SAME AS THE PREVIOUS WORKS, AND WITH SIMULTANEOUS AREA AND WIRE-LENGTH OPTIMIZATION [HB*-TREE (AREA+WL)], BASED ON THE MCNC BENCHMARKS

Circuit	SP [8]		Seg. Tree [3]		TCG-S [15]		SP w. Dummy [17]		HB*-tree		HB*-tree (Area + WL)		
	Area (mm^2)	Time (s)	Area (mm^2)	Time (s)	Area (mm^2)	Time (s)	Area (mm^2)	Time (s)	Area (mm^2)	Time (s)	Area (mm^2)	HPWL (mm)	Time (s)
apte	48.12	25	47.52	11	47.52	3	46.92	13	46.92	2	47.90	10.20	3
hp	9.84	138	9.71	62	9.71	50	9.43	13	9.35	2	10.10	30.74	16
ami33	1.24	684	1.23	307	1.21	423	1.24	23	1.23	12	1.29	47.23	39
ami49	37.82	2038	37.31	983	37.04	1247	38.32	29	36.85	20	41.32	769.99	96
Comparison	1.03	-	1.02	-	1.01	-	1.02	4.09	1.00	1.00	-	-	-

TABLE VI
COMPARISONS OF AREA UTILIZATION AND CPU TIMES FOR SP (ON SUN BLADE 100 500 MHz), SEGMENT TREE (SEG. TREE) (ON SUN BLADE 100 500 MHz), SP+LP (PENTIUM4 3.2 GHz), SP WITH DUMMY NODES (SP w. DUMMY) (ON PENTIUM4 3.2 GHz), AND HB*-TREE (ON PENTIUM4 3.2 GHz), BASED ON TWO REAL INDUSTRY BENCHMARKS

Circuit	SP [8]		Seg. Tree [12]		SP+LP [16]		SP w. Dummy [17]		HB*-tree		
	Area ($10^3 \mu m^2$)	Time (s)	Area ($10^3 \mu m^2$)	Time (s)	Area ($10^3 \mu m^2$)	Time (s)	Area ($10^3 \mu m^2$)	Time (s)	Area w/o Mod. Rot. ($10^3 \mu m^2$)	Area w/ Mod. Rot. ($10^3 \mu m^2$)	Time (s)
biasynth_2p4g	5.40	780	5.40	246	4.96	206	5.57	134	5.15	4.92	22
lnamixbias_2p4g	50.80	2824	50.30	726	50.15	3027	52.21	227	50.28	48.63	43
Comparison	1.071	-	1.066	-	1.016	39.88	1.103	5.68	1.040	1	1

("Area"), and the runtimes ("Time") for the aforementioned works and our HB*-tree with area optimization alone, same as the previous works, and with simultaneous area and wire-length optimization. The results of the works in [3], [8], and [15] are taken from [15], and those of [17] are based on the package provided by the authors. The results show that our HB*-tree achieves average area reductions of 3%, 2%, 1%, and 2% over [3], [8], [15], and [17], respectively. Note that the improvements should not be considered marginal, since the previous works have pushed the solution quality close to their limits. The main reason for the area improvement over the previous works is that our approach benefits from both the symmetry-island formulation and the short packing time of the proposed floorplan representations. Based on the symmetry-island formulation, the undesired solutions are pruned, and thus, we do not waste the time to search inferior solutions during SA. With the short packing time, we can search for more solutions within the same time limit. Consequently, our approach has greater possibility to find better solutions in shorter running time. For the running time, our algorithm is about 4.09× faster than in [17]. Since all other previous works ran on different platforms, it is not easy to report the speedups of our algorithm. Nevertheless, it is obvious from the table that our algorithm runs much faster than the previous works.

In the second set of experiments, we compared our algorithm with SPs in [8], segment trees in [12], SPs with linear programming in [16], and SPs with dummy nodes in [17]. Table VI lists the names of the industry benchmark circuits, the total areas, and the runtime for SPs, segment trees, SPs with linear programming, SPs with dummy nodes, and HB*-tree. The results show that our algorithm achieved average area reductions of 7.1%, 6.6%, 1.6%, and 10.3% over [8], [12], [16], and [17], respectively. In some applications, the orientations of analog device modules may not be allowed to be changed. To make fair comparisons with the previous works, we also performed our algorithm without module rotation. Our results show only 2.4% and 4% area overheads without the rotation, compared to the results of SPs with linear programming [16] and our approach, respectively. For the running time, our approach achieves significant speedups over the previous works, which is about 39.88× and 5.68× faster than those in [16] and [17], respectively. Again, the previous works [8],

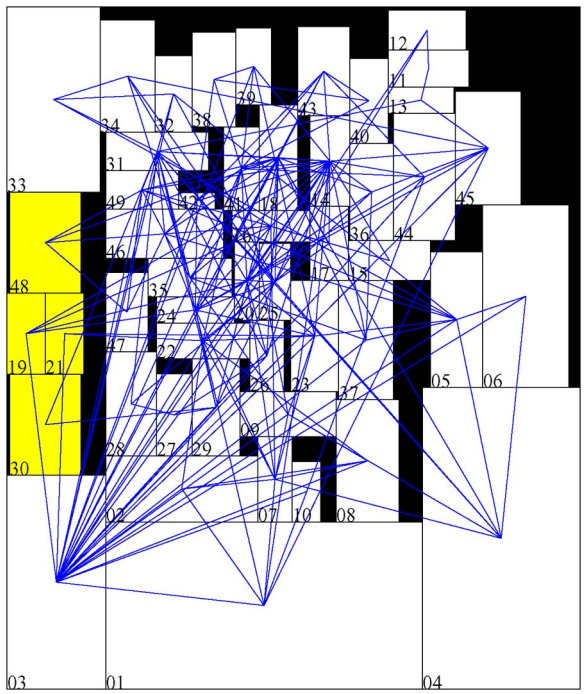
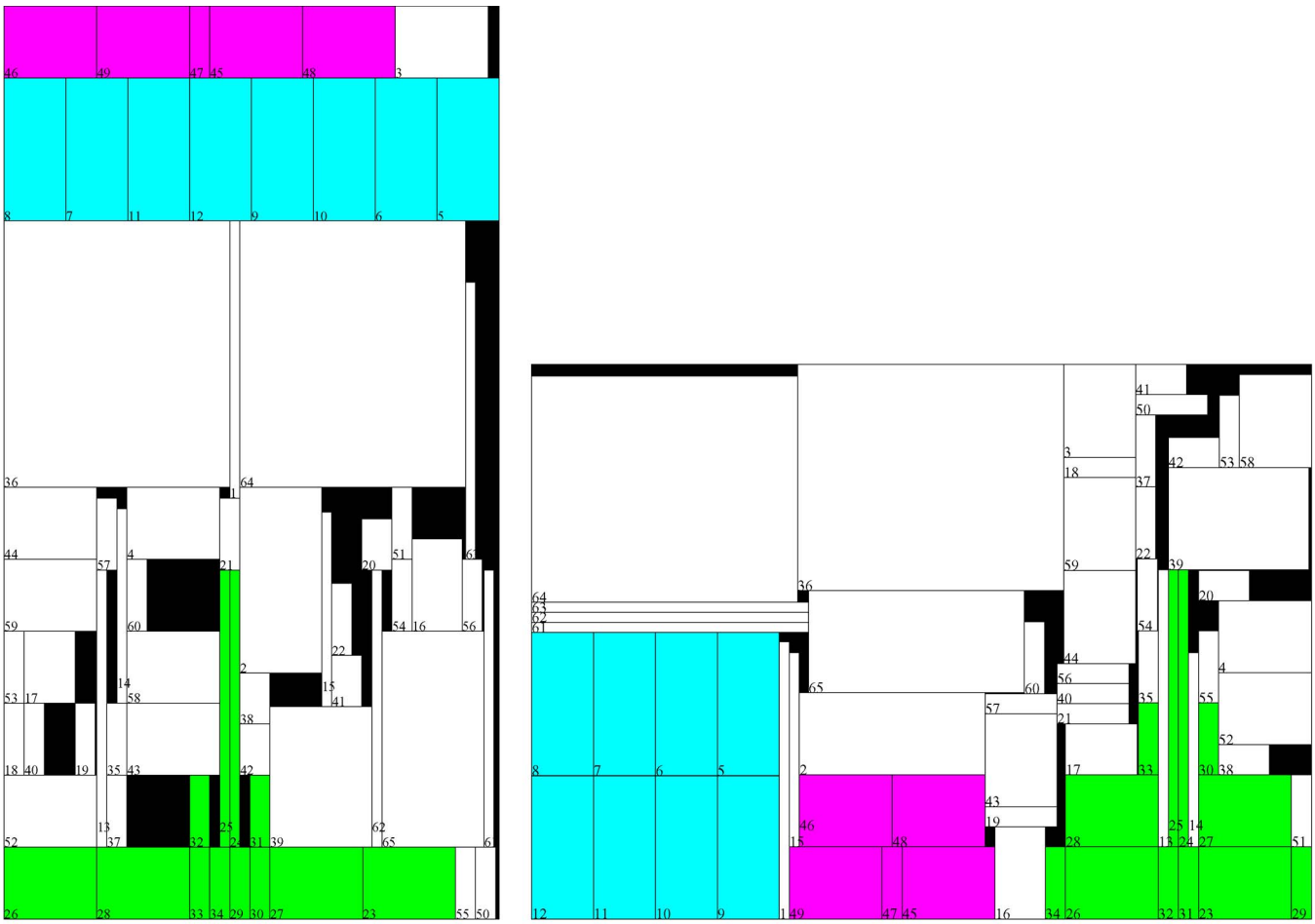


Fig. 19. Resulting placement of ami49 with simultaneous area and wire-length optimization, which contains the symmetry group, $S = \{(b_{19}, b_{21}), b_{30}^s, b_{48}^s\}$.



(a) (b)

Fig. 20. Resulting placement of biasynth_2p4g with three symmetry groups. (a) Resulting placement without module rotation. (b) Resulting placement with module rotation.

[12] ran on different platforms, and thus, we do not report the corresponding speedups; yet, it is obvious that our algorithm runs much faster than the previous works. It is clear from the two experiments that our algorithm achieves the best quality and efficiency than all published works.

Fig. 19 shows the resulting placement of ami49 with simultaneous area and wire-length optimization, which contains the symmetry group $S = \{(b_{19}, b_{21}), b^s_{30}, b^s_{48}\}$. Fig. 20 shows the resulting placements of biasynth_2p4g with the symmetry modules being colored.

VII. Conclusion and Future Work

We have proposed the first linear-time-packing algorithm for the placement with symmetry constraints, based on the symmetry-island formulation that prunes the solution subspace formed with nonsymmetry-island placements. We have introduced the concept of symmetry islands and presented the ASF-B*-trees to directly model the placement of a symmetry island. We have also presented the hierarchical HB*-trees to simultaneously optimize the placement with both symmetry islands and nonsymmetric modules. Experimental results have shown that our approach achieves the best-published quality and runtime efficiency for analog placement.

Acknowledgment

The authors would like to thank Prof. E. F. Y. Young and Y.-C. Tam of the Chinese University of Hong Kong for providing the package of their work [17] for the comparative studies.

References

[1] P.-H. Lin and S.-C. Lin, "Analog placement based on novel symmetry-island formulation," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2007, pp. 465–470.

[2] J. Cohn, D. Garrod, R. Rutenbar, and L. Carley, "KOAN/ANAGRAM II: New tools for device-level analog placement and routing," *IEEE J. Solid-State Circuits*, vol. 26, no. 3, pp. 330–342, Mar. 1991.

[3] F. Balasa, S. Maruvada, and K. Krishnamoorthy, "Efficient solution space exploration based on segment trees in analog placement with symmetry constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2002, pp. 497–502.

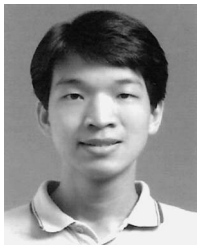
[4] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[5] D. Jepsen and C. Gelatt, Jr., "Macro placement by Monte Carlo annealing," in *Proc. IEEE Int. Conf. Comput. Des.*, Nov. 1983, pp. 495–498.

[6] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli, "Automation of IC layout with analog constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 8, pp. 923–942, Aug. 1996.

[7] K. Lampaert, G. Gielen, and W. Sansen, "A performance-driven placement tool for analog integrated circuits," *IEEE J. Solid-State Circuits*, vol. 30, no. 7, pp. 773–780, Jul. 1995.

[8] F. Balasa and K. Lampaert, "Symmetry within the sequence-pair representation in the context of placement for analog design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 7, pp. 721–731, Jul. 2000.
[9] Y. Pang, F. Balasa, K. Lampaert, and C.-K. Cheng, "Block placement with symmetry constraints based on the o-tree non-slicing representation," in *Proc. ACM/IEEE Des. Autom. Conf.*, 2000, pp. 464–467.
[10] F. Balasa, "Modeling non-slicing floorplans with binary trees," in *Proc. IEEE/ACM Int. Conf. Compu.-Aided Des.*, 2000, pp. 13–16.
[11] 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.*, 2000, pp. 458–463.
[12] F. Balasa, S. Maruvada, and K. Krishnamoorthy, "On the exploration of the solution space in analog placement with symmetry constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 2, pp. 177–191, Feb. 2004.
[13] F. Balasa, S. Maruvada, and K. Krishnamoorthy, "Using red–black interval trees in device-level analog placement with symmetry constraints," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Jan. 2003, pp. 777–782.
[14] S. Maruvada, A. Berkman, K. Krishnamoorthy, and F. Balasa, "Deterministic skip lists in analog topological placement," in *Proc. IEEE Int. Conf. ASIC*, Oct. 2005, vol. 2, pp. 834–837.
[15] J.-M. Lin, G.-M. Wu, Y.-W. Chang, and J.-H. Chuang, "Placement with symmetry constraints for analog layout design using TCG-S," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Jan. 2005, vol. 2, pp. 1135–1138.
[16] S. Koda, C. Kodama, and K. Fujiyoshi, "Linear programming-based cell placement with symmetry constraints for analog IC layout," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 4, pp. 659–668, Apr. 2007.
[17] Y.-C. Tam, Y. Young, and C. Chu, "Analog placement with symmetry and other placement constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2006, pp. 349–354.
[18] K. Krishnamoorthy, S. Maruvada, and F. Balasa, "Topological placement with multiple symmetry groups of devices for analog layout design," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 2032–2035.
[19] L. Zhang, C.-J. R. Shi, and Y. Jiang, "Symmetry-aware placement with transitive closure graphs for analog layout design," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Mar. 2008, pp. 180–185.
[20] M. Pelgrom, A. Duinmaijer, and A. Welbers, "Matching properties of MOS transistors," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1439, Oct. 1989.
[21] J.-M. Lin, H.-E. Yi, and Y.-W. Chang, "Module placement with boundary constraints using B*-trees," *Proc. Inst. Elect. Eng.—Circuits, Devices Syst.*, vol. 149, no. 4, pp. 251–256, Aug. 2002.
[22] G.-M. Wu, Y.-C. Chang, and Y.-W. Chang, "Rectilinear block placement using B*-trees," *ACM Trans. Design Autom. Electron. Syst.*, vol. 8, no. 2, pp. 188–202, Apr. 2003.
[23] M.-C. Wu and Y.-W. Chang, "Placement with alignment and performance constraints using the b*-tree representation," in *Proc. IEEE Int. Conf. Comput. Des.*, Oct. 2004, pp. 568–571.
[24] P.-H. Lin and S.-C. Lin, "Analog placement based on hierarchical module clustering," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2008, pp. 50–55.



Po-Hung Lin received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1998 and 2000, respectively. He is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan.

From 2000 to 2007, he was with Springsoft, Inc., Hsinchu. In 2008, he was a Visiting Scholar in the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana. His research interests include analog design automation and very large scale integration physical synthesis.

Yao-Wen Chang (S'94–A'96–M'96) 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 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, NTU. He is currently also a Visiting Professor at Waseda University, Kitakyushu, Japan. He was with National Chiao Tung University (NCTU), Hsinchu, Taiwan from 1996 to 2001 and IBM T. J. Watson Research Center in the summer of 1994. His current research interests lie in VLSI physical design, design for manufacturability/reliability, and design automation for biochips. He has been working closely with industry in these areas. He has co-edited one textbook on EDA and coauthored one book on routing and over 150 ACM/IEEE conference/journal papers in these areas.

Dr. Chang is a winner of the 2009 ACM ISPD Clock Network Synthesis Contest, the 2008 ACM ISPD Global Routing Contest, and the 2006 ACM ISPD Placement Contest. He is a recipient of Best Paper Award at ICCD-95 and 12 Best Paper Award Nominations from DAC (four times), ICCAD (twice), ISPD (three times), ACM TODAES, ASP-DAC, and ICCD in the past eight years. He has received many research awards, such as the 2007 Outstanding Research Award, the inaugural 2005 First-Class Principal Investigator Award, and the 2004 Dr. Wu Ta You Memorial Award, all from National Science Council of Taiwan, and the 2004 MXIC Young Chair Professorship from the MXIC Corp, and excellent teaching awards from NTU (five times) and NCTU.

He is currently an associate editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (TCAD) and an editor of the *Journal of Information Science and Engineering (JISE)* and the *Journal of Electrical and Computer Engineering (JECE)*. He has served on the ICCAD Executive Committee, the ASP-DAC Steering Committee, the ACM/SIGDA Physical Design Technical Committee, the ACM ISPD and IEEE FPT Organizing Committees, and the technical program committees of ASP-DAC, DAC, DATE, FPL, FPT, GLSVLSI, ICCAD, ICCD, IECON, ISPD, SOCC, TENCON, and VLSI-DAT. He is currently an independent board director of Genesys Logic, Inc., a technical consultant of RealTek Semiconductor Corp., a member of board of governors of Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.

Shyh-Chang Lin received the B.S. degree in control engineering from National Chiao Tung University, Hsinchu, Taiwan, 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 Physical Design Group, Springsoft, Inc., Hsinchu. His research interests include analog layout automation and physical design automation for very large scale integration.