

# Chapter 2

## Hierarchical Placement with Layout Constraints

Mark Po-Hung Lin and Yao-Wen Chang

**Abstract** In analog layout design, devices are required to be placed with matching, symmetry, and proximity constraints to reduce parasitic coupling effects and improve circuit performance. In addition to these basic placement constraints, there exist hierarchical symmetry and hierarchical proximity constraints due to circuit and layout design hierarchies. This chapter first introduces the hierarchical constraints induced by circuit and layout design hierarchies, and then presents a hierarchical placement approach to better consider these hierarchical constraints and effectively reduce the search space.

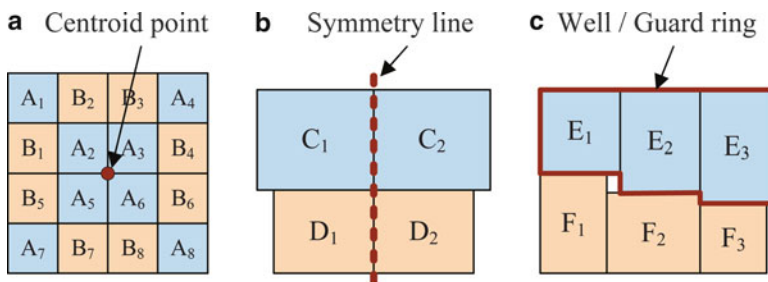
### 2.1 Introduction

According to [8, 11], the basic analog layout constraints include *common-centroid*, *symmetry*, and *proximity* constraints, illustrated in Fig. 2.1. The common-centroid constraint is usually applied to a subcircuit of a current mirror or a differential pair to reduce process-induced mismatches among the devices. The symmetry constraint is always required in the layout design of the whole differential subcircuit. It helps reduce the parasitic mismatches between two identical signal flows in the differential subcircuit. The proximity constraint is widely used in the subcircuit of a common device model or a certain circuit functionality. It helps form a connected placement of a subcircuit so that the subcircuit can share a connected substrate/well region or be surrounded by a common guard ring to reduce the layout area, the interconnecting wire length, and the substrate coupling effect. In particular, the placement outline of each subcircuit with the proximity constraint can be irregularly rectilinear for better area utilization. Figure 2.1c shows an example placement of two subcircuits,  $\{E_1, E_2, E_3\}$  and  $\{F_1, F_2, F_3\}$ , with the proximity constraint.

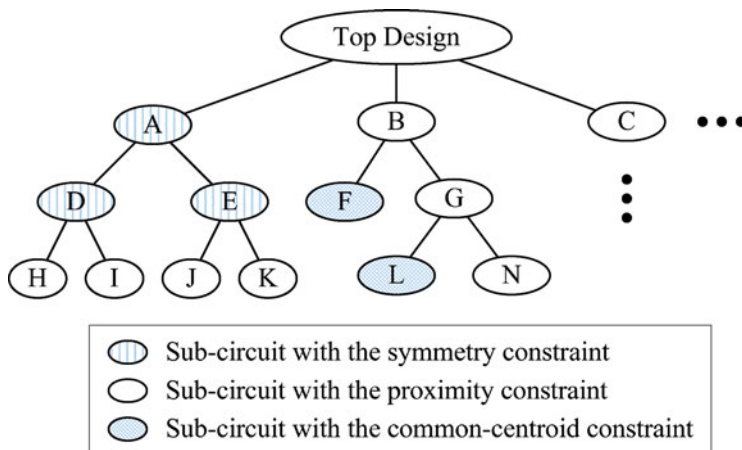
---

M.P.-H. Lin (✉)

Department of Electrical Engineering, National Chung Cheng University, Chiayi, Taiwan  
e-mail: [marklin@ccu.edu.tw](mailto:marklin@ccu.edu.tw)



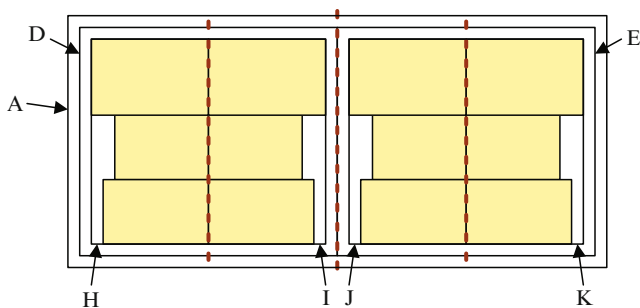
**Fig. 2.1** Basic analog layout constraints (a) common-centroid constraint (b) symmetry constraint (c) proximity constraint



**Fig. 2.2** Layout design hierarchy and the corresponding constraint in each subcircuit

Besides the basic layout constraints, there exist *hierarchical symmetry* and *hierarchical proximity* constraints due to *layout design hierarchy*. The layout design hierarchy may contain both exact and virtual hierarchies of analog circuit design. The exact hierarchy is the same as the circuit hierarchy, while the virtual hierarchy consists of hierarchical clusters [20]. Each cluster contains some devices and subcircuits, which are gathered based on device models, subcircuit functionality [10, 27], and/or other specific constraints [7]. Figure 2.2 shows an example layout design hierarchy, where each subcircuit corresponds to a specific constraint.

In Fig. 2.2, a subcircuit with the hierarchical symmetry constraint may contain some devices together with other subcircuits with the common-centroid and (hierarchical) symmetry constraints. Figure 2.3 shows an example hierarchical symmetric placement of several hierarchical subcircuits in Fig. 2.2. Similarly, a subcircuit with the hierarchical proximity constraint may contain some devices together with other subcircuits with the common-centroid, (hierarchical) symmetry, and (hierarchical) proximity constraints.



**Fig. 2.3** An example placement of the subcircuits  $A$ ,  $D$ ,  $E$ ,  $H$ ,  $I$ ,  $J$ , and  $K$  with the hierarchical symmetry constraint in Fig. 2.2, where the subcircuits  $H$  and  $I$  are symmetric,  $J$  and  $k$  are symmetric, and  $D$  and  $E$  are also symmetric

Based on the concept of the layout design hierarchy illustrated in Fig. 2.2, it is essential to synthesize analog layout hierarchically for better efficiency and effectiveness. It is also desirable to reduce the large search space by considering layout design hierarchy. Modern analog placement techniques often simultaneously optimize the placement in different hierarchical subcircuits, e.g., [17, 19, 20, 23, 24, 30], instead of bottom-up integration, because the optimal placement of a subcircuit may not lead to the globally optimal placement. Most of them apply simulated annealing [13] based on the topological floorplan representations, such as Sequence-Pair [28] and B\*-tree [6], while the latest one [30] adopts a fully deterministic approach. Among these works, the one based on the hierarchical B\*-tree (HB\*-tree) in [19, 20, 23] discussed how to handle the hierarchical symmetry and hierarchical proximity constraints together with the consideration of layout design hierarchy.

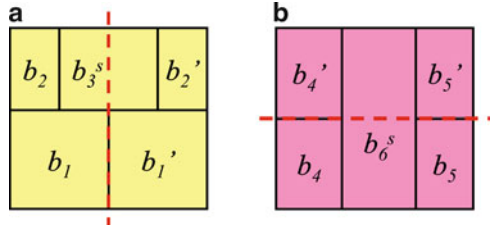
In the following sections, the basic hierarchical framework based on the HB\*-tree and symmetry-island formulation is first introduced to handle symmetry constraints. The generalized HB\*-tree is then presented to consider both hierarchical symmetry and hierarchical proximity constraints.

## 2.2 Preliminaries

### 2.2.1 Symmetry Constraints

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. The 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

**Fig. 2.4** Two symmetry types **(a)** symmetric placement with the vertical symmetry axis **(b)** symmetric placement with the horizontal symmetry axis



**Table 2.1** The notations in this chapter

$b$	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 $S_i$

with respect to a certain symmetry type. A symmetry type may correspond to a symmetry axis in either the horizontal or the vertical direction. Figure 2.4 shows two different symmetry types with either the vertical or the horizontal symmetry axis.

For the symmetric placement with the vertical (horizontal) symmetry axis shown in Fig. 2.4a (Fig. 2.4b), 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 structure is self-symmetric must have its center placed at the symmetry axis.

The notations listed in Table 2.1 are used throughout this chapter. Let  $S = \{S_1, S_2, \dots, S_m\}$  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  $\hat{y}_i$ ),  $1 \leq i \leq n$ . A symmetry group  $S_i = \{(b_1, b'_1), (b_2, b'_2), \dots, (b_p, b'_p), b_1^s, b_2^s, \dots, b_q^s\}$  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  $(b_j, b'_j)$ , and  $(x_k^s, y_k^s)$  denote 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 (2.1) [(2.2)].

$$\begin{aligned}
 x_j + x'_j &= 2 \times \hat{x}_i, & \forall j &= 1, 2, \dots, p. \\
 y_j &= y'_j, & \forall j &= 1, 2, \dots, p. \\
 x_k^s &= \hat{x}_i, & \forall k &= 1, 2, \dots, q.
 \end{aligned} \tag{2.1}$$

$$\begin{aligned}
x_j &= x'_j, & \forall j &= 1, 2, \dots, p. \\
y_j + y'_j &= 2 \times \hat{y}_i, & \forall j &= 1, 2, \dots, p. \\
y^s_k &= \hat{y}_i, & \forall k &= 1, 2, \dots, q.
\end{aligned} \tag{2.2}$$

### 2.2.2 Symmetry Island

Before introducing the symmetry island, the effect of the symmetric device layout on the electrical matching properties of the symmetric devices should be investigated. Pelgrom et al. [29] measured the mismatch between MOS transistors with various electrical parameters as a function of device areas, distances, and orientations. According to [29], the difference of an electrical parameter  $P$  between two rectangular devices is modeled by the standard deviation as shown in (2.3), 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  $D_x$ .

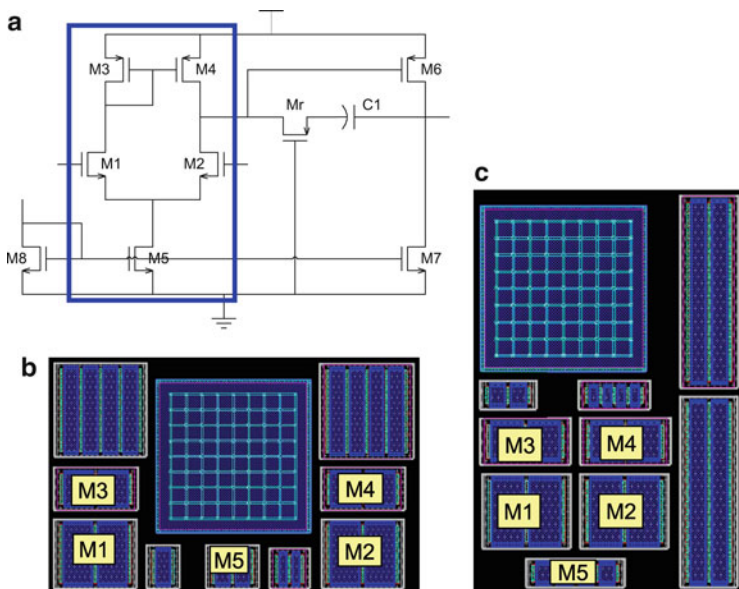
$$\sigma^2(\Delta P) = \frac{A_P^2}{WL} + S_P^2 D_x^2. \tag{2.3}$$

The device dimensions of modules in a symmetry pair are assumed to be 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. Figure 2.5a shows an analog circuit of a two-stage CMOS operational amplifier containing the differential input sub-circuit. The devices M1, M2, M3, M4, and M5 in the differential input sub-circuit form a symmetry group  $S = \{(M1, M2), (M3, M4), M5\}$ . Figures 2.5b, c show two corresponding layouts with different placement styles for the symmetry group  $S$ . The layout style in Fig. 2.5c is generally considered much better than that in Fig. 2.5b 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.

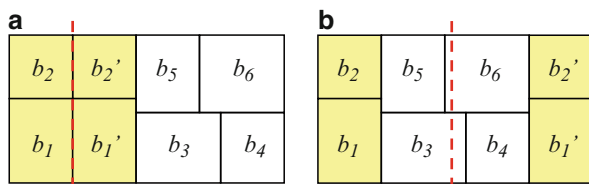
Based on the placement with the closest proximity for a symmetry group as shown in Fig. 2.5c, the concept of symmetry islands is then introduced with its definition given as follows:

**Definition 2.1.** 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.

In the example of Fig. 2.6, the symmetry group  $S_1$  in Fig. 2.6a forms a symmetry island, but that in Fig. 2.6b does not since it results in two disconnected components. The placement style in Fig. 2.6a is preferred in analog layout design due to its better electrical properties.



**Fig. 2.5** An example analog circuit and two different layout styles for the circuit. **(a)** The schematic of a two-stage CMOS operational amplifier, where the differential input sub-circuit forms a symmetry group. **(b)** A 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

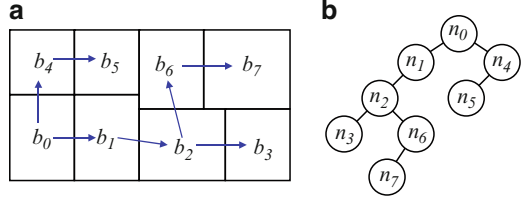


**Fig. 2.6** Two symmetric-placement examples of a symmetry group  $S_1 = \{(b_1, b_1'), (b_2, b_2')\}$ . **(a)**  $S_1$  forms a symmetry island. **(b)**  $S_1$  cannot form a symmetry island

### 2.2.3 Review of B\*-Trees

A B\*-tree is an ordered binary tree representing a *compacted placement*, in which every module cannot move left and bottom anymore. As shown in Fig. 2.7, 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.

**Fig. 2.7** (a) A compacted placement (same as Fig. 2.6a). (b) The B\*-tree representing the compacted placement in (a)



Given a B\*-tree, we can calculate the coordinate of each module by a pre-order 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 then its right subtree, this preorder tree traversal procedure, a.k.a. 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.

## 2.3 Placement of a Symmetry Group

### 2.3.1 Automatically Symmetric-Feasible B\*-tree

To consider the symmetric placement of a symmetry group and the packing of the symmetry modules to make a symmetry island, the automatically symmetric-feasible B\*-tree (ASF-B\*-tree for short) is proposed. Like 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.

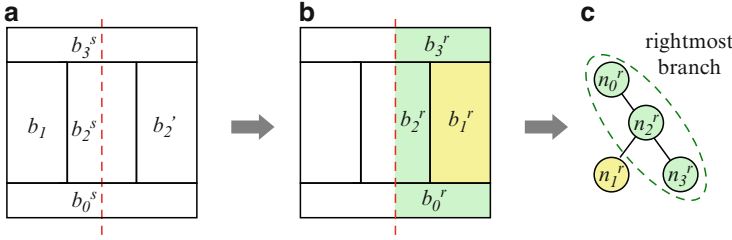
Before introducing the ASF-B\*-tree, the *representative* of a symmetry pair, the representative of a self-symmetric module, and the *representative B\*-tree* are defined as follows:

**Definition 2.2.** The representative  $b'_j$  of a symmetry pair  $(b_j, b'_j)$  is  $b'_j$ .

**Definition 2.3.** The representative  $b'_k$  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 of Fig. 2.8, the representative  $b'_1$  of the symmetry pair  $\{b_1, b'_1\}$  is  $b'_1$ , while the representative  $b'_0$  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



**Fig. 2.8** (a) A placement example of a symmetry group have a vertical symmetry axis. (b) Selecting a representative for each symmetry pair and self-symmetric module. (c) The 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

symmetry group should be the same as the number of symmetry pairs and self-symmetric modules. The representative B\*-tree is then defined as follows:

**Definition 2.4.** A representative B\*-tree is a B\*-tree containing only the representative nodes that correspond to representative modules.

Before explaining how to obtain an ASF-B\*-tree by making a representative B\*-tree symmetric-feasible for symmetric placements with vertical and horizontal symmetry axes, the *mirrored placement* of the representative modules for a symmetry group is introduced and defined as follows:

**Definition 2.5.** 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.

Based on the definition of the mirrored placement of the representative modules, the symmetric-feasible condition of a representative B\*-tree for the symmetric placements can be further defined as follows:

**Definition 2.6.** 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. 2.8a, 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, the representative module of each symmetry pair and self-symmetric module should be selected with the consideration of the placement on the right-half plane. Figure 2.8b highlights the representative modules, and Fig. 2.8c 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 symmetric-feasible, the following lemmas are derived, which formulate the symmetry conditions for self-symmetric modules and symmetry pairs.

**Lemma 2.1.** *The representative of a self-symmetric module must about 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 (2.1), 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 2.3, 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 about the symmetry axis  $\hat{x}$ . The case for a symmetry group with a horizontal symmetry axis can be proved similarly.

**Lemma 2.2.** *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  $b'$ .

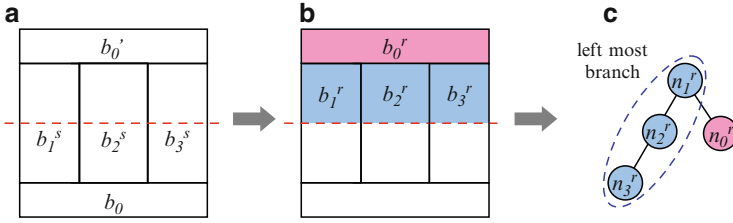
Given the coordinate of the representative  $b'$  and the vertical symmetry axis  $\hat{x}$ , the coordinate of the symmetric module  $b$  can be calculated by (2.1). We have  $x = 2 \times \hat{x} - x'$  and  $y = y'$ . After transposing  $\hat{x}$  to the left side and having the absolute value on both sides, we have  $|x - \hat{x}| = |\hat{x} - x'|$ . Since the representative is not on the symmetry axis, we have  $|x - \hat{x}| = |\hat{x} - x'| \geq \frac{w}{2}$ . 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.

According to Lemma 2.1 and the boundary constraints [21] in the B\*-trees, the symmetric-feasible representative B\*-trees have the following property:

**Property 2.1.** 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 right (left) most 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. 2.8c.

Similarly, the symmetric-feasible representative B\*-tree of the symmetric placement when the symmetry axis is in the horizontal direction can be derived. In this



**Fig. 2.9** (a) 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) The 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

case, we only consider the top-half plane during the placement of the representative modules. Figure 2.9c 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. 2.9a. Again, the representatives of the self-symmetric modules should about 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 illustrated in Fig. 2.9c.

Based on Definition 2.4 and Property 2.1, an ASF-B\*-tree is defined as follows:

**Definition 2.7.** An ASF-B\*-tree is a representative B\*-tree, which satisfies Property 2.1.

Once an ASF-B\*-tree is packed, the coordinates of these representatives are obtained, and the coordinates of their symmetric modules can be further calculated based on (2.1) and (2.2) with the given coordinates of the symmetry axes,  $\hat{x}_i$  and  $\hat{y}_i$ . The symmetric placement of a symmetry group automatically forms a symmetry island.

Based on Lemmas 2.1 and 2.2, the following theorems are derived:

**Theorem 2.1.** 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 2.1 and 2.2. 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.

**Theorem 2.2.** 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 (2.1) and (2.2). The symmetric modules also form a connected placement, and the boundary of the connected placement also about the symmetry axis (axes). Therefore, the whole symmetry group form 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.

**Theorem 2.3.** *There exists a unique correspondence between a compacted symmetric placement of a symmetry group and its induced ASF-B\*-tree.*

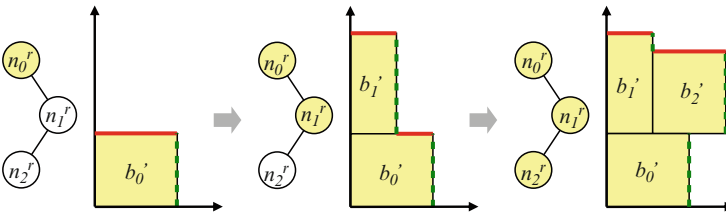
*Proof.* According to [6], there is a unique correspondence between an admissible placement and its induced B\*-tree. After obtaining the placement of the representative modules, the mirrored placement of the symmetric ones is also obtained. The mirrored placement is also unique. Therefore, there exists a unique correspondence between a compacted symmetric placement of a symmetry group and its induced ASF-B\*-tree.

Based on the above theorems, a corresponding symmetric placement for an ASF-B\*-tree can correctly and efficiently be found by avoiding searching in redundant solution spaces. It will be clear later in Sect. 2.6 that these advantageous properties of ASF-B\*-trees lead to superior solution quality and efficiency for analog placement.

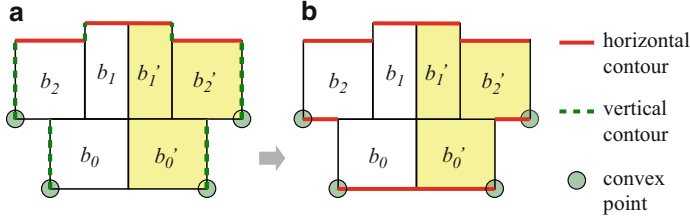
### 2.3.2 ASF-B\*-Tree Packing

The packing of the ASF-B\*-tree is similar to that of the B\*-tree [6], 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. Figure 2.10 shows the packing procedure of the example ASF-B\*-tree in Fig. 2.13a. 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, the coordinates of the symmetric modules and the extended contours can be



**Fig. 2.10** The packing procedure including the contour updates of the ASF-B\*-tree in Fig. 2.13a



**Fig. 2.11** The generation of the bottom contour of the symmetry island based on the dual vertical contours. (a) The convex points obtained by traversing the dual vertical contours from *bottom* to *top*. (b) The *bottom* horizontal contour connected by the convex points

calculated based on either (2.1) or (2.2). Figure 2.13b shows the resulting placement of the symmetry group and the contours of the symmetry island for the ASF-B\*-tree shown in Fig. 2.13a. As shown in Fig. 2.13b, the symmetry island contains one top horizontal and dual vertical contours. To further calculate the bottom horizontal contour of the symmetry island, both vertical contours from bottom need to be traversed to top and keep the convex points as shown in Fig. 2.11a. By connecting the convex points horizontally, the bottom horizontal contour of the symmetry island can be obtained as shown in Fig. 2.11b.

## 2.4 The Hierarchical Framework

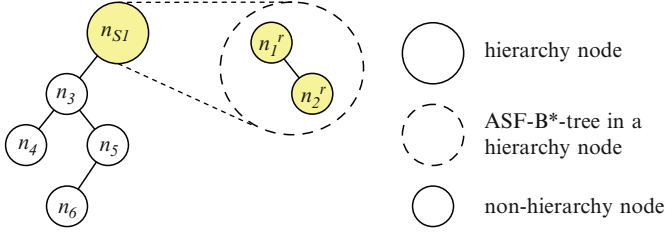
### 2.4.1 Hierarchical HB\*-Tree

The hierarchical framework, called *hierarchical B\*-tree* (HB\*-tree for short), is proposed 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.

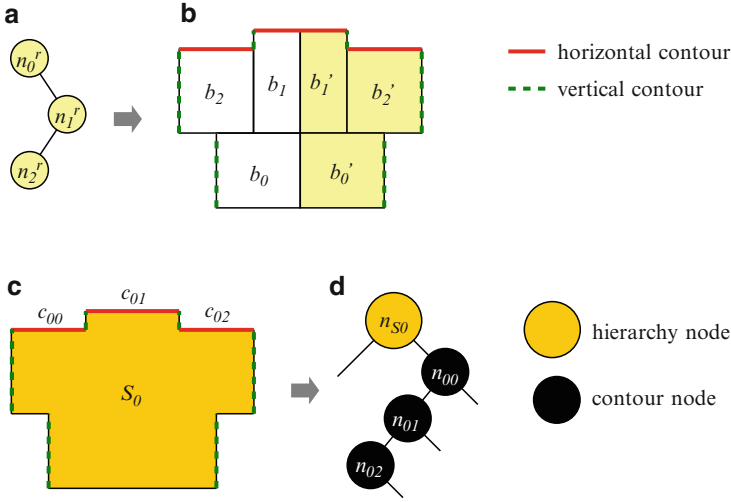
Figure 2.12 shows an HB\*-tree for the placement in Fig. 2.6a. Two symmetry groups,  $S_1$  and  $S_2$ , are represented by two hierarchy nodes,  $n_{S_1}$  and  $n_{S_2}$ , and each hierarchy node contains an ASF-B\*-tree that corresponds to a symmetry island in the symmetric placement.

### 2.4.2 HB\*-Tree with Rectilinear Symmetry Islands

The symmetry islands are often not rectangular, but are of rectilinear shapes. For example, in Fig. 2.13c, the symmetry island of the symmetry group  $S_0$  is of the rectilinear shape. Therefore, the HB\*-tree in Fig. 2.12 should be augmented to handle rectilinear symmetry islands. Wu et al. [33] proposed a method to deal with



**Fig. 2.12** An HB\*-tree for the placement in Fig. 2.6a



**Fig. 2.13** (a) An ASF-B\*-tree of a symmetry group  $S_0$ . (b) The horizontal and vertical contours of the corresponding placement. (c) The symmetry island and its effective contours. (d) The HB\*-tree for the rectilinear symmetry island

rectilinear modules by slicing a rectilinear module into several rectangular sub-modules 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, several *contour nodes* are introduced to represent top horizontal contour segments of the symmetry island. In Fig. 2.13c, there are three horizontal contour segments,  $c_{00}$ ,  $c_{01}$ , and  $c_{02}$ . The HB\*-tree is augmented by introducing the three contour nodes,  $n_{00}$ ,  $n_{01}$ ,  $n_{02}$ , as shown in Fig. 2.13d. 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 2.2.* 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 children of a regular module node must be a noncontour node.
5. The right child of a contour node, if any, must be a noncontour node.
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  $i$ th top contour segment of  $b_{S_0}$  from left to right.

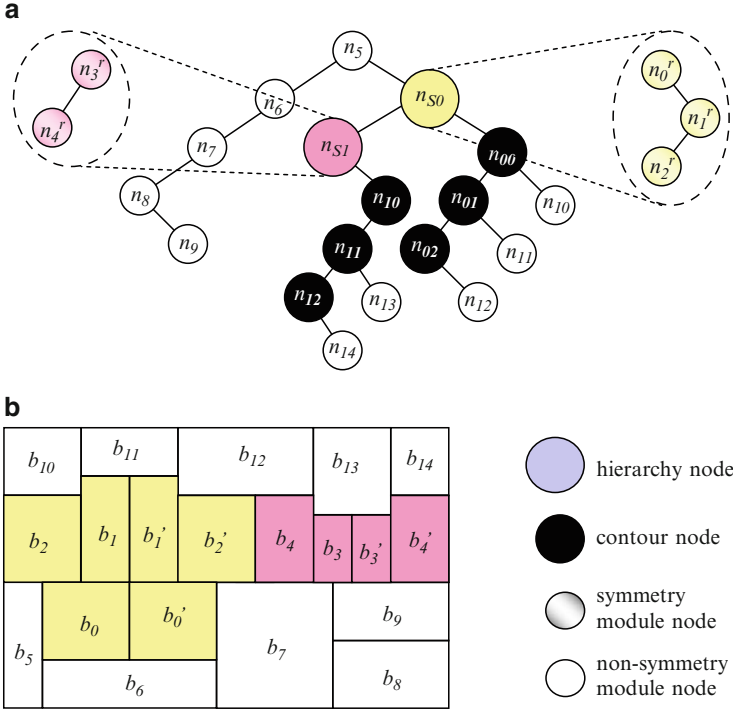
1. Since the contour node  $n_{0i}$  represents the  $i$ th top contour segment of  $b_{S_0}$ , it is impossible for  $n_{0i}$  to be the left child of  $n_{S_0}$  that corresponds to the lowest, adjacent module on the right side of  $b_{S_0}$ , based on the B\*-tree definition. The property thus follows.
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  $i$ th 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. 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.
5. The right child of the contour node  $n_{0i}$  represents the first module above the  $i$ th 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. This is a contradiction.
6. Based on the construction of the HB\*-tree, the parent of a contour node is either a contour node or a hierarchy node.

Figure 2.13a 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. 2.13b, the horizontal and vertical contours are obtained from the rectilinear outline after packing the ASF-B\*-tree. Figure 2.13c 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. 2.13d.

### 2.4.3 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, the best packing coordinate for the bottom boundary of the symmetry island should be calculated based on the bottom contour shown in Fig. 2.11b. The left child of the hierarchy node is then proceeded to be packed. After the left child and all its descendants are packed, the first contour node of the symmetry island is packed, followed by the second one, and so on. When packing the contour nodes, their corresponding coordinates should be updated and the hierarchy node should be replaced in the contour data structure of the HB\*-tree.

Figure 2.14a 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,  $n_5, n_6, n_7, n_8, n_9$ , are packed in the DFS order. The temporal contour list



**Fig. 2.14** (a) An HB\*-tree representing 20 modules with two symmetry groups  $S_0$  and  $S_1$ . (b) The resulting placement after packing the HB\*-tree

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, the resulting placement of the HB\*-tree is obtained as shown in Fig. 2.14b finally. Although the purpose of the packing is to obtain a compacted placement, sufficient white space might need to be allocated 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.

The following theorem shows the packing complexity.

**Theorem 2.4.** *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.

First, the packing for the ASF-B\*-tree of the symmetry group  $S_i$  in a hierarchy node is considered. 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 [6], the packing for a B\*-trees 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, the packing for the HB\*-tree is considered. 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, especially 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  $O(n)$ .

## 2.5 The Algorithm

The placement algorithm is based on simulated annealing [13]. Given a set of modules and symmetry constraints as the inputs, an initial solution represented by an HB\*-tree is constructed and then perturbed to search for a desired configuration until a predefined termination condition is satisfied. The cost function,  $\Phi(P)$ , of the placement is defined in (2.4), where  $\alpha$  and  $\beta$  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 (HPWL).

$$\Phi(P) = \alpha \times A_P + \beta \times W_P. \quad (2.4)$$

### 2.5.1 HB\*-Tree Perturbation

The following operations are applied to perturb an HB\*-tree.

- Op1: Rotate a module.
- Op2: Move a node to another place.
- Op3: Swap two nodes.

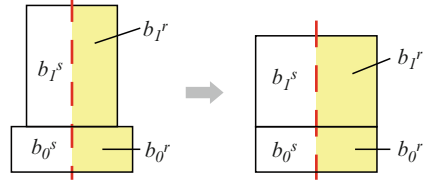
In the perturbation, the nonhierarchy nodes have higher probabilities to be selected because rotating, moving, or swapping the hierarchy nodes might incur a big jump in finding the next solution. It is well known that such a big jump might deteriorate the solution quality during the SA process. It should be noted that, due to the special structure of the HB\*-tree, a non-hierarchy node cannot be moved 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.

### 2.5.2 ASF-B\*-Tree Perturbation

In addition to the aforementioned Op1, Op2, and Op3 for HB\*-tree perturbation, the operations, Op4 and Op5, are introduced to perturb the ASF-B\*-trees. It should be noted that Property 2.1 should always be satisfied when perturbing an ASF-B\*-tree according to the definition of the ASF-B\*-trees in Definition 2.7.

- Op4: Change a representative.
- Op5: Convert a symmetry type.

**Fig. 2.15** 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  $b_1^r$



### 2.5.2.1 Module Rotation

When rotating modules in a symmetry group, the corresponding ASF-B\*-tree is unchanged. Two cases of symmetry-module rotation should be considered.

- 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. 2.15.

### 2.5.2.2 Node Movement

When moving a node to another place in an ASF-B\*-tree, the following two cases should be considered.

- 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, the representative node of a symmetry pair can be moved to anywhere in an ASF-B\*-tree. In Case 2, however, the representative node of a self-symmetric module can only be moved along the rightmost (leftmost) branch of the ASF-B\*-tree for vertical (horizontal) symmetric placement so that Property 2.1 is satisfied.

### 2.5.2.3 Node Swapping

When swapping two nodes in an ASF-B\*-tree, the following two cases should be considered.

- 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, two nodes representing the representatives of two different symmetry pairs can be arbitrarily swapped. However, 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 2.1 is still satisfied after node swapping.

#### 2.5.2.4 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. The representative of either a symmetry pair or a self-symmetric module can be changed.

- Case 1: Change the representative of a symmetry pair.
- Case 2: Change the representative of a self-symmetric module.

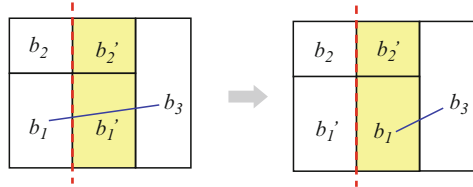
In Case 1, for a symmetry pair  $(b_j, b'_j)$ , the representative can be changed from  $b_j$  to  $b'_j$  or from  $b'_j$  to  $b_j$ . Figure 2.16 illustrates that changing the representative of the symmetry pair  $(b_1, b'_1)$  from  $b'_1$  to  $b_1$  may result in shorter wire length between  $b_1$  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 illustrated in Fig. 2.17, 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.

#### 2.5.2.5 Symmetry-Type Conversion

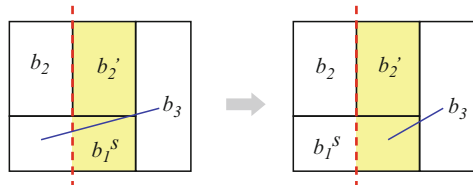
For symmetry-type conversion of a symmetry group, both conversions between the vertical symmetry and the horizontal one should be considered.

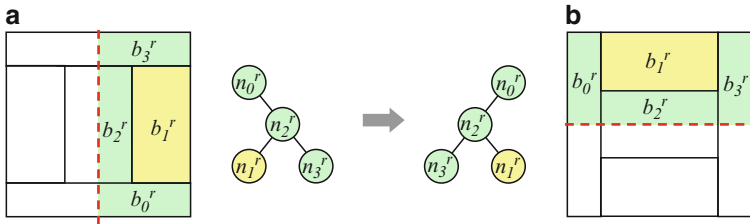
- Case 1: Convert the symmetry type from vertical symmetry to horizontal one.
- Case 2: Convert the symmetry type from horizontal symmetry to vertical one.

**Fig. 2.16** 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  $b_3$

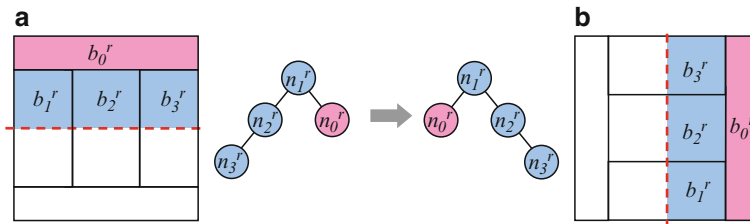


**Fig. 2.17** Changing the representative of the self-symmetric module  $b_1^s$  may result in shorter wire length between  $b_1^s$  and  $b_3$





**Fig. 2.18** Converting the symmetry type from (a) vertical symmetry to (b) horizontal symmetry



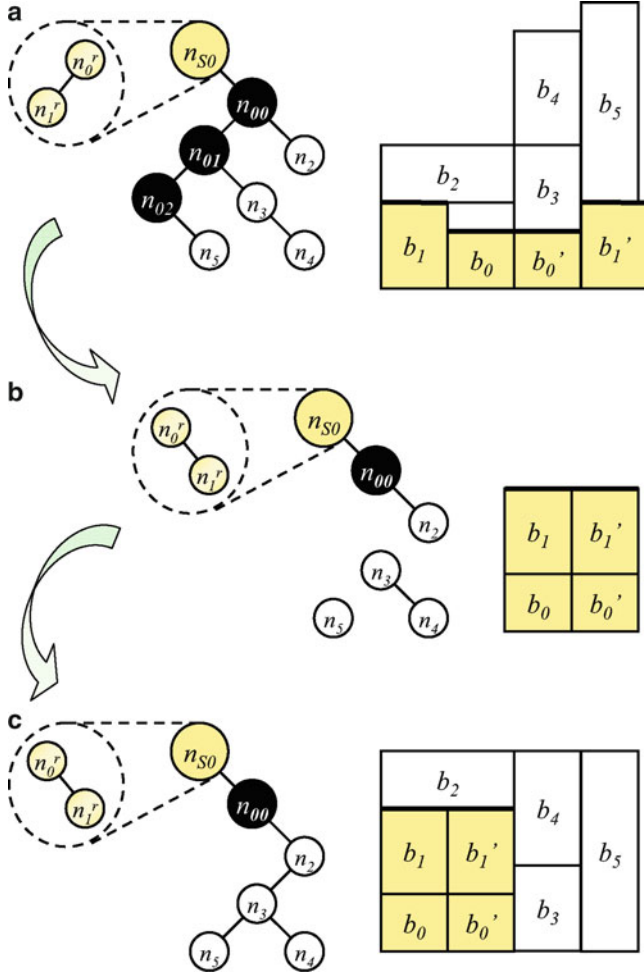
**Fig. 2.19** Converting the symmetry type from (a) horizontal symmetry to (b) vertical symmetry

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. Figures 2.18 and 2.19 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.

### 2.5.3 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. Such nodes, called *dangling nodes* should be reassigned to new parents. To keep the relative placement topology before and after perturbing an ASF-B\*-tree, the nearest contour node is searched 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, the leftmost-skewed child of the right child is traversed.



**Fig. 2.20** An example of updating contour-related nodes. **(a)** An HB\*-tree and its corresponding placement containing the symmetry group  $S_0 = \{(b_0, b_0'), (b_1, b_1')\}$ . **(b)** The intermediate HB\*-tree after perturbing the ASF-B\*-tree in the hierarchy node  $n_{S_0}$ , and the corresponding symmetry island of  $S_0$ . The contour-related nodes,  $n_3$  and  $n_5$ , become dangling. **(c)** The HB\*-tree after updating the contour-related nodes and its corresponding placement

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.

Figure 2.20 shows an example of updating contour-related nodes. In Fig. 2.20a, 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  $n_0^r$ . The placement of  $S_0$  forms

a new symmetry island as shown in Fig. 2.20b, 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  $n_3$ . Finally,  $n_3$  is assigned to be the parent of  $n_5$ , and  $n_5$  is assigned as the left child of  $n_3$ .

## 2.6 Comparisons with Other Approaches

In this section, we compare existing topological analog placement methods considering symmetry constraints, based on theoretical and empirical studies. The first subsection explores the time complexities of the perturbation and packing operations adopted by the existing topological methods, and the second subsection conducts experiments based on the simulated annealing algorithm and two sets of commonly used benchmarks.

### 2.6.1 Comparisons of Time Complexities

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 [13] in combination with floorplan representations to handle symmetry constraints. These representations can be classified into two major categories: (1) the absolute representation and (2) the topological representation.

For the absolute representation first proposed by Jepsen and Gellat [12], 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 [9], PUPPY-A [25], and LAYLA [16] 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. The most popular floorplan representations include the B\*-tree [6], Sequence Pair (SP) [28], and TCG [18].

For the B\*-tree representation, Balasa et al. derived its symmetric-feasible condition [1]. To explore the solution space in the symmetric-feasible B\*-trees, they augmented the B\*-tree [6] using various data structures, including segment trees [3, 5], red-black trees [4], and deterministic skip lists [26], to reduce the packing time.

**Table 2.2** Comparisons of popular analog placement approaches considering symmetry constraints based on topological floorplan representations.  $n$ : the number of modules;  $m$ : the number of symmetry pairs

Analog placement approach considering symmetry constraints	Perturbation time	Packing time
B*-tree [1]	$O(\lg n)$	$O(n^2)$
B*-tree + Seg. tree [3]	$O(\lg n)$	$O(n \lg n)$
BT + RB-tree [4]	$O(\lg n)$	$O(n \lg n)$
BT + Skip list [26]	$O(\lg n)$	$O(n \lg n)$
Sequence-pair (SP) [2]	$O(1)$	$O(n^2)$
SP + LP [14]	$O(1)$	$\Omega(n^2)$
SP w. dummy [31]	$O(1)$	$O(n^2)$
SP w. priority queue [15]	$O(1)$	$O(m \cdot n \lg \lg n)$
TCG-S [22]	$O(n^2)$	$O(n^2)$
TCG [34]	$O(n)$	$O(n^2)$
B*-tree w. ESF + LP [30]	N/A	$\Omega(n^2)$
ASF-B*-tree + HB*-tree	$O(\lg n)$	$O(n)$

More recently, Strasser et al. [30] proposed a deterministic approach based on the B\*-tree representation [6] with enhanced shape functions (ESF) and linear programming (LP).

For the SP representation, Balasa et al. also derived its symmetric-feasible condition [2]. By taking advantage of the symmetry-feasible condition, Koda et al. [14] proposed a linear programming based method. Tam et al. [31] introduced a dummy node and additional constraint edges for each symmetry group after obtaining a symmetric-feasible sequence pair. Krishamoorthy et al. [15] 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.

For the TCG representation, Lin et al. presented its symmetric-feasible conditions [22]. However, it requires  $O(n^2)$  time to perturb and pack TCGs. Zhang et al. [34] further improved the perturbation time of the TCG representation from  $O(n^2)$  to  $O(n)$ .

Table 2.2 compares aforementioned analog placement approaches considering symmetry constraints based on topological floorplan representations. It should be noted that “ASF-B\*-tree + HB\*-tree” is the fastest algorithm among all these popular approaches, while “SP + LP [14]” and “B\*-tree w. ESF + LP [30]” take at least  $\Omega(n^2)$  packing time due to LP. Since the approach [30] explores all placement configurations for a small set of device modules and groups in each hierarchy, the perturbation of the B\*-tree is not required.

### 2.6.2 Comparisons of Experimental Results

The placement algorithms were implemented in the C++ programming language on a 3.2GHz Intel Pentium4 PC under the Linux operation system. Two sets of

**Table 2.3** MCNC benchmark circuits

Circuit	# of mod.	# of sym. mod.	Mod. area (mm <sup>2</sup> )
apte	9	8	46.56
hp	11	8	8.83
ami33	33	6	1.16
ami49	49	4	35.45

**Table 2.4** Industry benchmark circuits

Circuit	# of mod.	# of sym. mod.	Mod. area (10 <sup>3</sup> μm <sup>2</sup> )
biasynth_2p4g	65	8 + 12 + 5	4.70
lnamixbias_2p4g	110	16 + 6 + 6 + 12 + 4	46.00

experiments were performed: one is based on the four MCNC benchmarks (apte, hp, ami33, and ami49) used in [22], and the other consists of two real industry analog designs (biasynth\_2p4g and lnamixbias\_2p4g) used in [5] and [14]. (Note that they both were extracted by Koda et al. [14] from Figs. 9 and 10 in [5].) Table 2.3 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 2.4 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”).

Based on simulated annealing, a left-skewed HB\*-tree was constructed as the initial solution. The initial temperature  $T_0$  was calculated by (2.5), where  $\Delta_{\text{avg}}$  is the average uphill cost and  $P$  is the initial probability to accept uphill solutions. During the simulated annealing 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.

$$T_0 = -\Delta_{\text{avg}} / \ln P. \quad (2.5)$$

In the first set of experiments, the HB\*-tree is compared with the following works: sequence pairs [2], segment trees [3], TCG-S [22], and sequence pairs with dummy nodes [31]. Table 2.5 lists the names of the MCNC benchmark circuits (“Circuit”), the total areas (“Area”), and the runtimes (“Time”) for the aforementioned works and the HB\*-tree with area optimization alone, same as the other works, and with simultaneous area and wirelength optimization. The results of the works [2, 3, 22] are taken from the paper [22], and those of [31] are based on the package provided by the authors. The results show that the HB\*-tree achieves average area reductions of 3%, 2%, 1%, and 2% over [2], [3], [22], and [31], respectively. Noted that the improvements should not be considered marginal since the other works have pushed the solution quality close to their limits. The main reason for the area improvement over the other works is that the HB\*-tree 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 the time is saved to search inferior solutions during simulated

**Table 2.5** Comparisons of area utilization and CPU times for sequence pair (SP) (on Sun Sparc Ultra-60 433MHz), segment tree (seg. tree) (on Sun Sparc Ultra-60 433MHz), TCG-S (on Sun Sparc Ultra-60 433MHz), sequence pair with dummy nodes (SP w. dummy) (on Pentium4 3.2GHz), and our HB\*-tree (on Pentium4 3.2GHz) with area optimization alone, same as the previous works, and with simultaneous area and wirelength optimization (HB\*-tree (area + WL)), based on the MCNC benchmarks

Circuit	SP [2]		Seg. tree [3]		TCG-S [22]		SP w. dummy [31]		HB*-tree		HB*-tree (area + WL)		
	Area (mm <sup>2</sup> )	Time (s)	Area (mm <sup>2</sup> )	Time (s)	Area (mm <sup>2</sup> )	Time (s)	Area (mm <sup>2</sup> )	Time (s)	Area (mm <sup>2</sup> )	Time (s)	Area (mm <sup>2</sup> )	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	2,038	37.31	983	37.04	1,247	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	–	–	–

annealing. With the short packing time, it is possible to search for more solutions within the same time limit. Consequently, the HB\*-tree has greater possibility to find better solutions in shorter running time. For the running time, the HB\*-tree is approximately  $4.09\times$  faster than [31]. Since all the other works ran on different platforms, it is not easy to report the speedups of our algorithm. Nevertheless, it is obvious from the table that the HB\*-tree runs much faster than the other works.

In the second set of experiments, the HB\*-tree is compared with sequence pairs in [2], segment trees in [5], sequence pairs with linear programming in [14], and sequence pairs with dummy nodes in [31]. Table 2.6 lists the names of the industry benchmark circuits, the total areas and the runtime for sequence-pairs, segment trees, sequence-pairs with linear programming, sequence-pairs with dummy nodes, and HB\*-tree. The results show that the HB\*-tree achieved average area reductions of 7.1%, 6.6%, 1.6%, and 10.3% over [2], [5], [14], and [31], respectively. In some applications, the orientations of analog device modules may not be allowed to be changed. To make fair comparisons with the other works, the HB\*-tree was also performed without module rotation. The results show only 2.4% and 4% area overheads without the rotation, compared to the results of sequence pairs with linear programming [14] and the HB\*-tree, respectively. For the running time, the HB\*-tree achieves significant speedups over the other works, which is approximately  $39.88\times$  and  $5.68\times$  faster than those in [14] and [31], respectively. Again, the other works [2, 5] ran on different platforms, and thus the corresponding speedups are not reported, yet it is obvious that the HB\*-tree runs much faster than the previous works. It is clear from the two experiments that the HB\*-tree achieves the best quality and efficiency than all the other works.

Figure 2.21 shows the resulting placement of ami49 with simultaneous area and wirelength optimization, which contains the symmetry group  $S = \{(b_{19}, b_{21}), b_{30}^s, b_{48}^s\}$ . Figure 2.22 shows the resulting placements of biasynth\_2p4g without module rotation, while Fig. 2.23 shows the resulting placements of biasynth\_2p4g with module rotation.

## 2.7 Advanced Symmetry Constraints

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

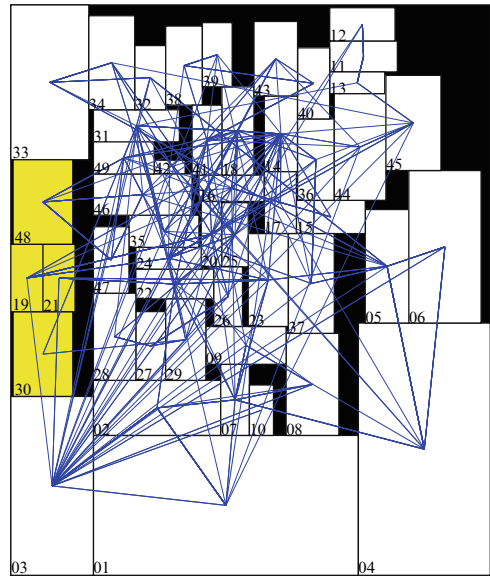
### 2.7.1 Multiple Symmetry-Group Alignment

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, a zero-height

**Table 2.6** Comparisons of area utilization and CPU times for sequence pair (SP) (on Sun Blade 100 500 MHz), segment tree (seg. tree) (on Sun Blade 100 500 MHz), SP+LP (Pentium4 3.2 GHz), sequence pair 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 [2]		Seg. tree [5]		SP+LP [14]		SP w. dummy [31]		HB*-tree	
	Area ( $10^3 \mu\text{m}^2$ )	Time (s)	Area ( $10^3 \mu\text{m}^2$ )	Time (s)	Area ( $10^3 \mu\text{m}^2$ )	Time (s)	Area ( $10^3 \mu\text{m}^2$ )	Time (s)	Area w/o mod. rot. ( $10^3 \mu\text{m}^2$ )	Time (s)
biasynth_2p4g	5.40	780	5.40	246	4.96	206	5.57	134	5.15	4.92
lnamixbias_2p4g	50.80	2,824	50.30	726	50.15	3,027	52.21	227	50.28	48.63
Comparison	1.071	–	1.066	–	1.016	39.88	1.103	5.68	1.040	1

**Fig. 2.21** The resulting placement of ami49 with simultaneous area and wirelength optimization, which contains the symmetry group,  $S = \{(b_{19}, b_{21}), b_{30}^s, b_{48}^s\}$



(zero-width) dummy block can be inserted right at the left (bottom) of each to-be-aligned symmetry island. A dummy node is then introduced 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 [32].

### 2.7.2 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. Figure 2.24 shows two examples, including the symmetric placements and the corresponding ASF-B\*-trees, which integrate nonsymmetric module clusters into symmetry groups. In Fig. 2.24a, 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. 2.24b, 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  $C_2'$ .

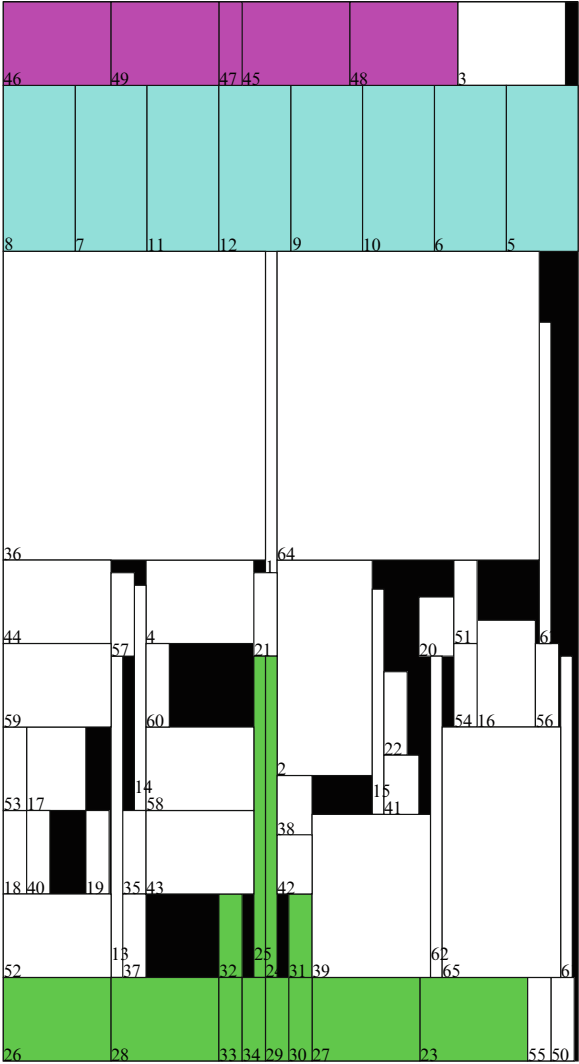


Fig. 2.22 The resulting placement of biasynth\_2p4g without module rotation

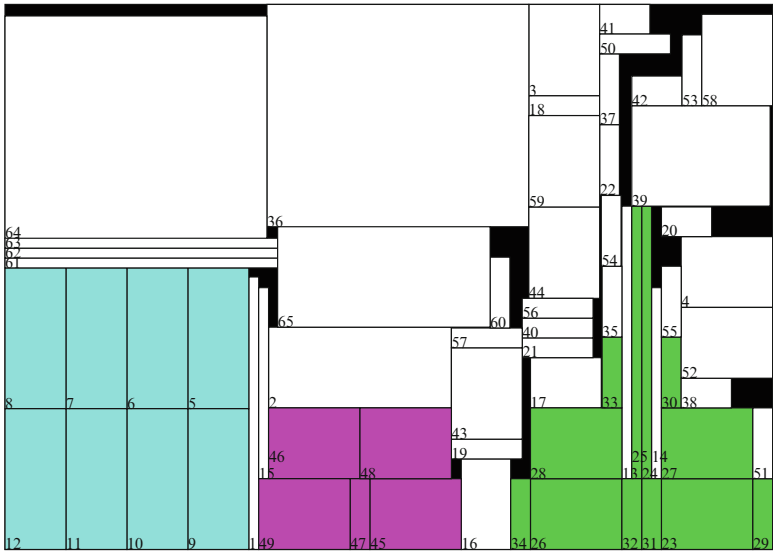


Fig. 2.23 The resulting placement of biasynth\_2p4g with module rotation

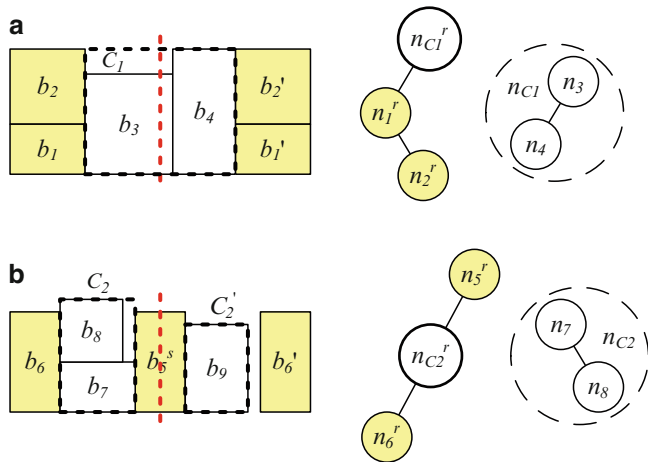


Fig. 2.24 Integrating nonsymmetric modules into symmetry groups. (a) The nonsymmetric modules form the self-symmetric module cluster  $C_1 = \{b_3, b_4\}$  in the symmetry group  $S_1 = \{(b_1, b_1'), (b_2, b_2'), C_1^s\}$ . (b) The nonsymmetric modules form two clusters,  $C_2 = \{b_7, b_8\}$  and  $C_2' = \{b_9\}$ , as a symmetry pair in the symmetry group  $S_2 = \{b_5^s, (b_6, b_6'), (C_2, C_2')\}$

## 2.8 Hierarchical Constraints

### 2.8.1 Hierarchical Symmetry

In some fully symmetric analog designs, such as the example in Fig. 2.3, 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 formulation, a hierarchical tree structure [20] 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.

### 2.8.2 Hierarchical Clustering/Proximity

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  $C = \{C_1, C_2, \dots, C_l\}$  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  $C_j$ ,  $C_i$  is called a super-cluster, and  $C_j$  is called a subcluster. The hierarchical clustering constraint limits all the device modules and/or subclusters of the same super-cluster to a connected placement.

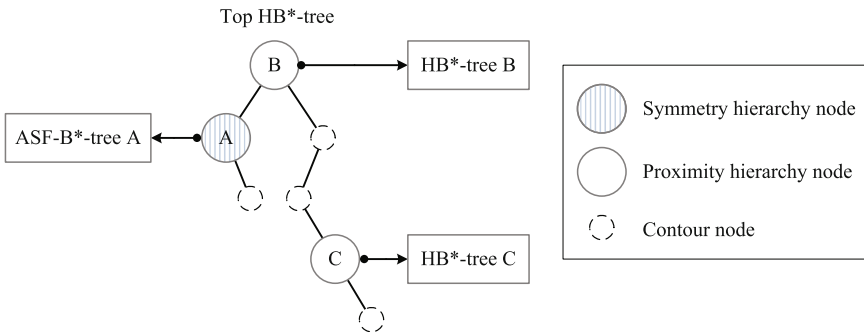


Fig. 2.25 Example HB\*-trees modeling the hierarchical floorplan of the design in Fig. 2.2

To formulate the hierarchical clustering constraint using the HB\*-trees, each of the hierarchy nodes  $n_{C_1}, n_{C_2}, \dots, n_{C_l}$  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.

Figure 2.25 shows example HB\*-trees modeling the hierarchical placement of the design in Fig. 2.2. Consequently, the number of the HB\*-trees will be equal to that of the subcircuits plus one for modeling the top design. When perturbing the HB\*-trees, one of the HB\*-trees should be selected first, and then any perturbation operation for the B\*-tree can be applied to the selected HB\*-tree. When converting the HB\*-trees to a hierarchical placement, the packing procedure is also similar to that for the B\*-tree, which adopts a preorder tree traversal. Once a hierarchy node is traversed, the nodes in the HB\*-tree linked by the hierarchy node will be traversed before traversing the next node in the HB\*-tree to which the hierarchy node belongs. During the HB\*-tree packing, the properties of the proximity constraint should also be considered [20].

The hierarchical framework based on the HB\*-tree can easily integrate other placement approaches for different subcircuits with different placement requirements. Besides integrating the ASF-B\*-tree, the HB\*-tree can also integrate both the corner block list (CBL) and the grid-based approach in [24] for a common-centroid placement, the signal-flow driven approach [17] for the placement of a specific subcircuit with clear signal flows, and other placement approaches.

## 2.9 Conclusion

This chapter has introduced hierarchical analog placement framework, HB\*-tree, with the consideration of layout design hierarchy and hierarchical placement constraints. Different from the existent approaches with at least log-linear-time algorithms, a linear-time packing algorithm has been presented based on the symmetry-island formulation that prunes the solution subspace formed with nonsymmetry-island placements. Experimental results have shown that such approach achieves high quality and runtime efficiency for analog placement.

## References

1. F. Balasa, Modeling non-slicing floorplans with binary trees, *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 13–16, 2000
2. F. Balasa and K. Lampart, Symmetry within the sequence-pair representation in the context of placement for analog design, *IEEE Trans. Computer-Aided Design*, 19(7):721–731, 2000

3. F. Balasa, S. Maruvada, and K. Krishnamoorthy, Efficient solution space exploration based on segment trees in analog placement with symmetry constraints, *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 497–502, Nov 2002
4. F. Balasa, S. Maruvada, and K. Krishnamoorthy, Using red-black interval trees in device-level analog placement with symmetry constraints, *Proceedings of IEEE/ACM Asia South Pacific Design Automation Conference*, pp. 777–782, Jan 2003
5. F. Balasa, S. Maruvada, and K. Krishnamoorthy, On the exploration of the solution space in analog placement with symmetry constraints, *IEEE Trans. Computer-Aided Design*, 23(2): 177–191, 2004
6. Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, B\*-trees: a new representation for non-slicing floorplans, *Proceedings of ACM/IEEE Design Automation Conference*, pp. 458–463, 2000
7. E. Charbon, E. Malavasi, and A. Sangiovanni-Vincentelli. Generalized constraint generation for analog circuit design, *Proceedings of ACM/IEEE International Conference on Computer-Aided Design*, 408–414, 1993
8. J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Charley. Analog Device-Level Layout Automation. *Kluwer, Dordrecht*, 1994
9. 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*, 26(3):330–342, 1991
10. H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich. The sizing rules method for analog integrated circuit design, *Proceedings of ACM/IEEE International Conference on Computer-Aided Design*, 343–349, 2001
11. A. Hastings. The Art of Analog Layout. 2nd Ed. *Pearson Prentice Hall*, 2006
12. D. Jepsen and C. Gelatt Jr., Macro placement by monte carlo annealing, *Proceedings of IEEE International Conference on Computer Design*, pp. 495–498, Nov 1983
13. S. Kirkpatrick, J. Gelatt, C. D., and M. P. Vecchi, Optimization by Simulated Annealing, *Science*, 220(4598):671–680, 1983
14. S. Koda, C. Kodama, and K. Fujiyoshi, Linear programming-based cell placement with symmetry constraints for analog ic layout, *IEEE Trans. Computer-Aided Design*, 26(4):659–668, 2007
15. K. Krishnamoorthy, S. Maruvada, and F. Balasa, Topological placement with multiple symmetry groups of devices for analog layout design, *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 2032–2035, May 2007
16. K. Lampaert, G. Gielen, and W. Sansen, A performance-driven placement tool for analog integrated circuits, *IEEE J. Solid-State Circuits*, 30(7):773–780, 1995
17. D. Long, X. Hong, and S. Dong. Signal-path driven partition and placement for analog circuit, *Proceedings of ACM/IEEE Asia South Pacific Design Automation Conference*, 694–699, 2006
18. J.-M. Lin and Y.-W. Chang, TCG: A transitive closure graph based representation for general floorplans, *IEEE Trans. VLSI Systems*, 13(2):288–292, 2005
19. P.-H. Lin and S.-C. Lin, Analog placement based on novel symmetry-island formulation, *Proceedings of ACM/IEEE Design Automation Conference*, pp. 465–470, Jun 2007
20. P.-H. Lin and S.-C. Lin, Analog placement based on hierarchical module clustering, *Proceedings of ACM/IEEE Design Automation Conference*, pp. 50–55, Jun 2008
21. J.-M. Lin, H.-E. Yi, and Y.-W. Chang, Module placement with boundary constraints using B\*-trees, *IEEE Proceedings – Circuits, Devices and Systems*, 149(4):251–256, 2002
22. J.-M. Lin, G.-M. Wu, Y.-W. Chang, and J.-H. Chuang, Placement with symmetry constraints for analog layout design using TCG-S, *Proceedings of IEEE/ACM Asia South Pacific Design Automation Conference*, vol. 2, pp. 1135–1138, Jan 2005
23. P.-H. Lin, Y.-W. Chang, and S.-C. Lin, Analog placement based on symmetry-island formulation, *IEEE Trans. Computer-Aided Design*, 28(6):791–804, 2009
24. Q. Ma, E. F. Y. Young, K. P. Pun. Analog placement with common centroid constraints, *Proceedings of ACM/IEEE International Conference on Computer-Aided Design*, 579–585, 2007
25. E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli, Automation of ic layout with analog constraints, *IEEE Trans. Computer-Aided Design*, 15(8):923–942, 1996

26. S. Maruvada, A. Berkman, K. Krishnamoorthy, and F. Balasa, Deterministic skip lists in analog topological placement, *Proceedings of IEEE International Conference on ASIC*, vol. 2, pp. 834–837, 2005
27. T. Massier, H. Graeb, and U. Schlichtmann. Sizing rules for bipolar analog circuit design, *Proceedings of ACM/IEEE International Conference on Design, Automation and Test in Europe*, 140–145, 2008
28. H. Murata, K. Fujiiyoshi, S. Nakatake, and Y. Kajitani. VLSI module placement based on rectangle-packing by the sequence-pair, *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 15(12):1518–1524, 1996
29. M. Pelgrom, A. Duinmaijer, and A. Welbers, Matching properties of mos transistors, *IEEE J. Solid-State Circuits*, 24(5):1433–1439, 1989
30. M. Strasser, M. Eick, H. Graeb, U. Schlichtmann, and F. M. Johannes. Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions, *Proceedings of ACM/IEEE International Conference on Computer-Aided Design*, 2008
31. Y.-C. Tam, E. F. Y. Young, and C. Chu, Analog placement with symmetry and other placement constraints, *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 349–354, Nov 2006
32. M.-C. Wu and Y.-W. Chang, Placement with alignment and performance constraints using the B\*-tree representation, *Proceedings of IEEE International Conference on Computer Design*, pp. 568–571, Oct 2004
33. G.-M. Wu, Y.-C. Chang, and Y.-W. Chang, Rectilinear block placement using B\*-trees, *ACM Transactions on Design Automation of Electronics Systems*, 8(2):188–202, 2003
34. L. Zhang, C.-J. R. Shi, and Y. Jiang, Symmetry-aware placement with transitive closure graphs for analog layout design, *Proceedings of IEEE/ACM Asia South Pacific Design Automation Conference*, pp. 180–185, Mar 2008

Analog Layout Synthesis

A Survey of Topological Approaches

Graeb, H.E. (Ed.)

2011, XV, 302 p., Hardcover

ISBN: 978-1-4419-6931-6