# Analog Placement Based on Novel Symmetry-Island Formulation

**Conference Paper** · June 2007

DOI: 10.1145/1278480.1278601 · Source: DBLP



**2 authors**, including:

Shyh-Chang Lin
National Formosa University
**7** PUBLICATIONS **263** CITATIONS

SEE PROFILE

# Analog Placement Based on Novel Symmetry-Island Formulation

Po-Hung Lin[†‡] and Shyh-Chang Lin[‡]
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan[†]
Research and Development Department, Springsoft, Inc., Hsinchu 300, Taiwan[‡]
marklin@eda.ee.ntu.edu.tw; chris_lin@springsoft.com

## ABSTRACT

In this paper, we present *the first* amortized linear-time packing algorithm for the placement with symmetry constraints. 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 B*-trees (ASF-B*-trees) to directly model the placement of a symmetry island. Unlike the previous works that can handle only 1D symmetry constraints, our ASF-B*-tree is *the first* in the literature to additionally consider 2D symmetry. We then present hierarchical B*-trees (HB*-trees) which can simultaneously optimize the placement with both symmetry islands and non-symmetry modules. Unlike the previous works, our approach can guarantee the close proximity of symmetry modules 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 amortized linear) and much faster than previous works which need at least loglinear time. Experimental results show that our approach achieves the best published quality and runtime efficiency for analog placement.

**Categories and Subject Descriptors:** B.7.2 [Integrated Circuits]: Design Aids - Layout, Placement and Routing

**General Terms:** Algorithms, Design

**Keywords:** Analog placement, symmetry, floorplanning

## 1. INTRODUCTION

For analog layout design, some pairs of modules need to be placed symmetrically with respect to one or two common axes. There are several advantages of the symmetric placement: It reduces the effect of parasitic mismatches which may lead to higher offset voltages and degrade power-supply rejection ratio [7]. It can also reduce the circuit sensitivity to thermal gradients or 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]. Further, the symmetric modules are usually placed at closest proximity for better electrical properties such as parasitic matching and thermal gradients.

### 1.1 Previous Work

The problem of analog placement considering symmetry constraints has been extensively studied in the literature 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 [1], O-tree [13], and binary trees [2]. To explore the solution space in the symmetric-feasible binary trees, they augmented the B*-tree [6] using various data structures, including segment trees [3, 5], red-black trees [4], and deterministic skip lists [12]. Lin et al. [11] also presented the symmetric-feasible conditions for the TCG-S representation. Two more recent works [9, 14] further took advantage of the symmetry-feasible condition in sequence pairs [1]. Kouda et al. [9] proposed a linear programming based method, and Tam et al. [14] introduced a dummy node and additional constraint edges for each symmetry group after obtaining a symmetric-feasible sequence pair. 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, Kouda et al. [9] 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 observe 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 simulated annealing (SA) [8] 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 need $\Omega(n \lg n)$ 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. Third, all previous works can handle only the symmetric placement with respect to only one axis. However, there exist some analog designs that require modules to be symmetric to two common axes simultaneously.

### 1.2 Our Contributions

In this paper, we present *the first* amortized linear-time packing algorithm for the placement with symmetry constraints, compared to the previous works that need $\Omega(n \lg n)$ time. Specifically, the packing complexity of the previous works [1, 2, 9, 11, 13, 14] are all $O(n^2)$ time while those of [3, 4, 5, 12] need $O(n \lg n)$ time.

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 [6], we propose a representation called *automatically symmetric-feasible B*-trees (ASF-B*-trees)* that can directly model the 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


*DAC'07,* June 4–7, 2007, San Diego, CA

rectilinear placement. It guarantees a symmetric placement and do not need to verify/fix the symmetric-feasible conditions during SA perturbations. In addition to 1D symmetry, we also explore the placement with 2D symmetry in the ASF-B*-tree. To our best knowledge, this is the *first work* in the literature to handle 2D symmetry constraints for analog placement.

We then present a hierarchical framework called hierarchical B*-trees (HB*-trees) which can simultaneously optimize the placement with both symmetry islands and non-symmetry 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 amortized linear) and much faster than previous works. Experimental results based on the MCNC benchmarks [11] and the real industry designs used in [9] show that our approach produces the best published results and runtime efficiency for analog placement. Further, the scalability of our approach is much better than those of the previous works.

The remainder of this paper is organized as follows. Section 2 gives the preliminaries about the symmetry constraints, symmetry islands, and the B*-tree representation. Section 3 presents how to model the placement of a symmetry group as a symmetry island using the ASF-B*-tree. Section 4 proposes the hierarchical framework, HB*-tree, and Section 5 presents our placement algorithm. Section 6 reports the experimental results, and finally Section 7 concludes this paper.

# 2. 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 [6] on which our work is based.

## 2.1 Symmetry Constraints

Symmetry constraints can be formulated in terms of *symmetry types*, *symmetry groups*, *symmetry pairs*, and *self-symmetry modules*. In analog layout design, there are three major symmetry types: 1D vertical symmetry, 1D horizontal symmetry, and 2D symmetry. Figure 1 shows the three symmetry types. The placement of a symmetry group in 1D vertical (horizontal) symmetry has the symmetry axis in the vertical (horizontal) direction, illustrated in Figure 1(a) (Figure 1(b)), while that in 2D symmetry has symmetry axes in both directions, as illustrated in Figure 1(c).

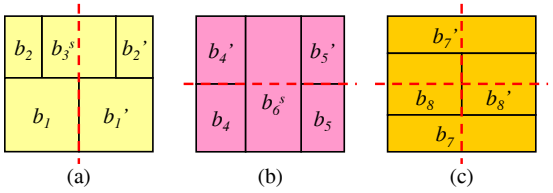



**Figure 1: Three symmetry types. (a) 1D vertical symmetry. (b) 1D horizontal symmetry. (c) 2D symmetry.**

Let $S = \{S_1, S_2, ..., S_n\}$ be a set of $n$ 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 $S_i = \{(b_1, b_1'), (b_2, b_2'), ..., (b_p, b_p'), b_1^s, b_2^s, ..., b_q^s\}$ consists of $p$ symmetry pairs and $q$ self-symmetry modules, where $(b_j, b_j')$ denotes a symmetry pair and $b_k^s$ denotes a self-symmetry module. A symmetry pair, containing two modules with the same dimensions and orientations, should be placed symmetrically along the symmetry axis (axes). A self-symmetry module must have its center point placed on the symmetry axis (axes). Let $(x_j, y_j)$ and $(x_j', y_j')$ denote the respective coordinates of the center points of two modules $b_j$ and $b_j'$ in a symmetry pair $(b_j, b_j')$, and $(x_k^s, y_k^s)$ denotes the coordinate of the center point of the self-symmetry module $b_k^s$. The symmetric placement of a symmetry group $S_i$ with 1D vertical (horizontal) symmetry must satisfy the three equations in Group (1) (Group (2)) listed below, while that with 2D symmetry must satisfy either of the four equations in Group (3) or Group (4).

| (1) | (2) | (3) | (4) |
|---|---|---|---|
| $2\times\hat{x}_i = x_j + x_j'$ | $x_j = x_j'$ | $2\times\hat{x}_i = x_j + x_j'$ | $\hat{x}_j = x_j = x_j'$ |
| $y_j = y_j$ | $2\times\hat{y}_i = y_j + y_j'$ | $\hat{y}_j = y_j = y_j'$ | $2\times\hat{y}_i = y_j + y_j$ |
| $\hat{x}_i = x_k^s$ | $\hat{y}_i = y_k^s$ | $\hat{x}_i = x_k^s$ | $\hat{x}_i = x_k^s$ |
| | | $\hat{y}_i = y_k^s$ | $\hat{y}_i = y_k^s$ |

$\forall i = 1,2,...,n. \quad \forall j = 1,2,...,p. \quad \forall k = 1,2,...,q.$

Figure 2(a) shows a symmetric placement example of eight modules, including one symmetry group, $S_1 = \{(b_1, b_1'), (b_2, b_2')\}$, and three non-symmetry modules, $b_3$, $b_4$, $b_5$, and $b_6$.

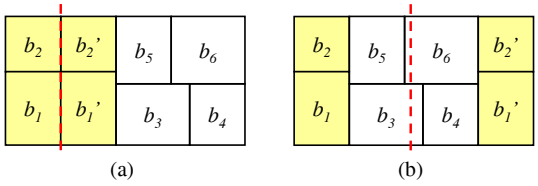



**Figure 2: Two symmetric-placement examples of a symmetry group $S_1 = \{(b_1, b_1'), (b_2, b_2')\}$. (a) $S_1$ forms a symmetry island. (b) $S_1$ cannot form a symmetry island.**

To reduce the circuit sensitivities due to thermal gradients and process variations, modules of the same symmetry group are usually placed at close proximity (or even adjacent) to each other. To obtain such a placement, 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 the modules in the symmetry group form a connected placement.*

In Figure 2(a), the symmetry group $S_1$ forms a symmetry island, but that in Figure 2(b) does not since it results in two disconnected components. The placement style in Figure 2(a) is usually preferred in analog layout design due to its better electrical properties.

## 2.2 Review of B*-trees

Since our work is based on the B*-tree representation [6], 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 cannot move left and bottom anymore. As shown in Figure 3, 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.

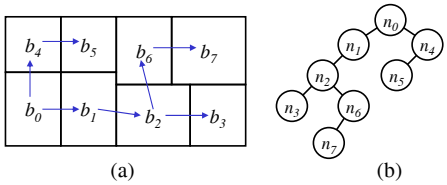



**Figure 3: (a) A compacted placement (same as Figure 2(a)). (b) The B*-tree representing the compacted placement in (a).**

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 pre-order 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 amortized linear to the number of modules.

# 3. PLACEMENT OF A SYMMETRY GROUP

In this section, we propose the automatically symmetric-feasible B*-tree (ASF-B*-tree for short) to consider the symmetric placement of a symmetry group and the packing of the symmetry modules to make a symmetry island. Besides, we also explore different symmetry types in ASF-B*-trees. Before introducing the ASF-B*-tree, we should define the representative of a symmetry pair or a self-symmetry module.

DEFINITION 2. *The representative $b_j^r$ of a symmetry pair ($b_j$, $b'_j$) is $b'_j$ (the right half of $b'_j$) in a 1D (2D) symmetric placement.*

For the example of 1D symmetry in Figure 4, the representative $b_1^r$ of the symmetry pair $\{b_1, b'_1\}$ is $b'_1$. Another example of 2D symmetry in Figure 5, the representative $b_0^r$ of the symmetry pair $b_0, b'_0$ is the right half of $b'_0$.

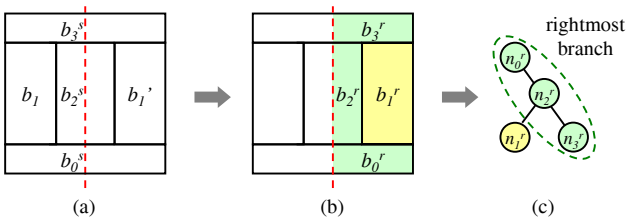



**Figure 4: (a) A placement example of a symmetry group with 1D vertical symmetry. (b) Selecting a representative for each symmetry pair and self-symmetry module. (c) The ASF-B*-tree representing the placement of the symmetry group, where the dash circled nodes represent the left-boundary modules.**

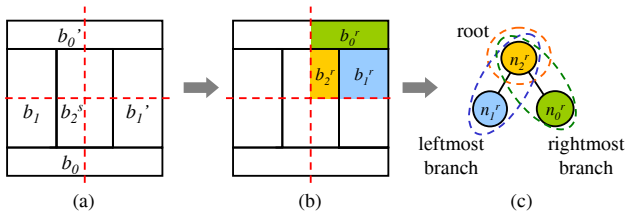



**Figure 5: (a) A placement example of a symmetry group with 2D symmetry. (b) Selecting a representative module for each symmetry pair and self-symmetry module. (c) The ASF-B*-tree representing the placement of the symmetry group, where the root represents the corner module, the rightmost-branch nodes represent the left-boundary modules, and the leftmost-branch nodes represent the bottom-boundary modules.**

DEFINITION 3. *The representative $b_k^r$ of a self-symmetry module $b_k^s$ is the right half (the top-right quarter) of $b_k^s$ in a 1D (2D) symmetric placement.*

For the example of 1D symmetry in Figure 4, the representative $b_0^r$ of the self-symmetry module $b_0^s$ is the right half of $b_0^s$. Another example of 2D symmetry in Figure 5, the representative $b_2^r$ of the self-symmetry module $b_2^s$ is the top-right quarter of $b_2^s$.

Note that each symmetry pair or self-symmetry 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-symmetry modules. Now, we can define an ASF-B*-tree as follows:

DEFINITION 4. *An ASF-B*-tree is a B*-tree containing only the representative nodes that correspond to the representative modules.*

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 equations in Group (1), (2), (3), or (4) 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. In the following, we introduce the ASF-B*-trees for both 1D and 2D symmetric placements.

## 3.1 1D Symmetric Placement

For a symmetry group in a 1D symmetric placement, the symmetry axis can be either vertical or horizontal. In Figure 4(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 ASF-B*-tree, we should select the representative module of each symmetry pair and self-symmetry module and consider the placement on the right-half plane. Figure 4(b) highlights the representative modules, and Figure 4(c) shows the corresponding ASF-B*-tree of the symmetric placement. Each node in the ASF-B*-tree corresponds to a representative module.

To make the ASF-B*-tree of 1D symmetry symmetric-feasible, the representative of a self-symmetry module must abut the symmetry axis. According to the boundary constraints [10] in the B*-trees, the nodes representing the modules on the left boundary should be on the rightmost branch as shown in Figure 4(c).

Similarly, we can get the ASF-B*-tree of the symmetric placement when the symmetry axis is in horizontal direction. In this case, we only consider the top-half plane during the placement of the representative modules.

## 3.2 2D Symmetric Placement

We can also represent the 2D symmetric placement of a symmetry group using the ASF-B*-tree. Figure 5(a) shows the 2D symmetric placement of the symmetry group $S = \{(b_0, b'_0), (b_1, b'_1), b_2^s\}$, where modules $b_0$ and $b'_0$ are placed symmetrically along the horizontal symmetry axis, modules $b_1$ and $b'_1$ are placed symmetrically along the vertical symmetry axis, and module $b_2$ is self-symmetric on both axes. For the 2D symmetric placement, we only need to consider the top-right quarter of the plane and select a representative module for each symmetry pair and self-symmetry module, as shown in Figure 5(b). Figure 5(c) shows the corresponding ASF-B*-tree.

Again, to make the ASF-B*-tree of a 2D symmetric placement automatically symmetric-feasible, the representative of a symmetry pair must abut one of the symmetry axes, and that of a self-symmetry module must abut both symmetry axes. Therefore, we have left-boundary, bottom-boundary, and corner constraints for a 2D symmetric placement. Each constraint is illustrated in

We have the following theorems for the proposed ASF-B*-trees:

THEOREM 1. *An ASF-B*-tree is symmetric-feasible in both 1D and 2D symmetric placements.*

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

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

Based on these 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 6 that these nice properties of ASF-B*-trees lead to superior solution quality and efficiency for analog placement.

# 4. THE 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 non-symmetry modules. In an HB*-tree, the symmetry island of each symmetry group can be in any rectilinear shapes, and symmetry and non-symmetry modules are simultaneously placed to optimize the placement.

## 4.1 HB*-tree Representation

Figure 6 shows an HB*-tree for the placement in Figure 2(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.

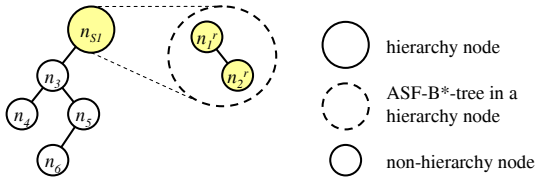



**Figure 6: An HB*-tree for the placement in Figure 2(a).**

The symmetry islands are often not rectangular, but are of rectilinear shapes. For example, in Figure 7(c), the symmetry island of the symmetry group $S_0$ is of the rectilinear shape. Therefore, we should augment the HB*-tree in Figure 6 to handle rectilinear symmetry islands. Wu et al. [15] proposed a method to deal with 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 sub-modules during B*-tree perturbations.

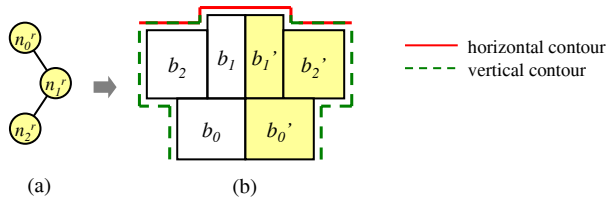
Instead of slicing a rectilinear symmetry island, we introduce *contour nodes* to represent top horizontal contour segments of the symmetry island. In Figure 7(c), there are three horizontal contour segments, $c_{00}$, $c_{01}$, and $c_{02}$. We augment the HB*-tree by introducing the three contour nodes, $n_{00}$, $n_{01}$, $n_{02}$, as shown in Figure 7(d). Each contour node keeps the coordinates of the corresponding horizontal contour segment.

Figure 7(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 Figure 7(b), the horizontal and vertical contours are obtained from the rectilinear outline after packing the ASF-B*-tree. Figure 7(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 Figure 7(d).

## 4.2 HB*-tree Packing

Similar to the B*-tree packing, the HB*-tree packing runs in the DFS order. Before packing an HB*-tree, the ASF-B*-tree in each hierarchy node should be packed first to obtain the outline of the symmetry island. The vertical and horizontal 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 effective two (dual) vertical contours shown in Figure 7(c). It takes amortized constant time to calculate the best packing coordinate of the symmetry island. After the left child of the hierarchy node 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 coordinates and replace the hierarchy node in the contour data structure of the HB*-tree.

Figure 8(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, $n_5$, $n_6$, $n_7$, $n_8$, $n_9$, are packed in the DFS order. The temporal contour list is $< n_5, n_6, n_7, n_9 >$. 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 $< n_{S_0}, n_7, n_9 >$. Continuing the packing procedure, we can obtain the resulting placement of the HB*-tree in Figure 8(b) finally.



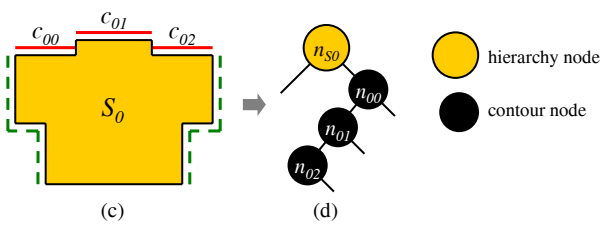




**Figure 7: (a) An ASF-B*-tree of a symmetry group $S_0$. (b) The horizontal and vertical contours of the corresponding placement. (c) The symmetry island and its effective contours. (d) The HB*-tree for the rectilinear symmetry island.**

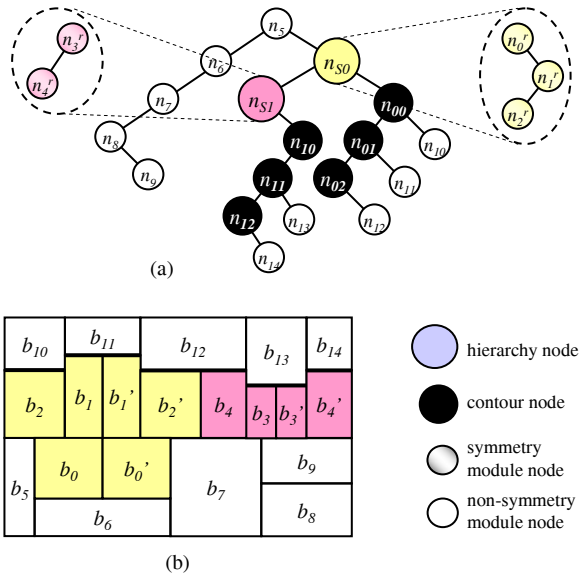



**Figure 8: (a) An HB*-tree representing 20 modules with two symmetry groups $S_0$ and $S_1$. (b) The resulting placement after packing the HB*-tree.**

# 5. THE ALGORITHM

Our algorithm is based on the simulated annealing [8]. Given an initial solution represented by an HB*-tree, we perturb it to search for a "good" configuration until a predefined termination condition is satisfied. The cost function, $\Phi(P)$, of the placement is defined in Equation (1), 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).

$$\Phi(P) = \alpha \times A_P + \beta \times W_P. \quad (1)$$

## 5.1 HB*-tree Perturbation

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

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

In the perturbation, the non-hierarchy 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 non-hierarchy 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.

## 5.2 ASF-B*-tree Perturbation

To perturb an ASF-B*-trees, we introduce two additional operations, Op4 and Op5, for the ASF-B*-trees in addition to the aforementioned Op1, Op2, and Op3.

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

We explain the operations in the following.

### 5.2.1 Representative Change

The purpose of changing a representative for a symmetry pair or a self-symmetry module is to optimize the wire length, while the area is kept unchanged after changing the representative. There are four cases to change the representative.

- Case 1: Change a representative of a symmetry pair in 1D symmetric placement.
- Case 2: Change a representative of a symmetry pair in 2D symmetric placement.
- Case 3: Change a representative of a self-symmetry module in 1D symmetric placement.
- Case 4: Change a representative of a self-symmetry module in 2D symmetric placement.

In Case 1, for a symmetry pair $(b_i, b_i')$, we can simply change the representative from $b_i$ to $b_i'$ or from $b_i'$ to $b_i$. In Case 2, there are four representatives for a symmetry pair, and two representatives for each module. The operation is to change the representative from the one in the current module to one of the representatives in the other module of the symmetry pair. For Cases 3 and 4, changing the representative of a self-symmetry module is similar to the operation of rotation or flip. Obviously, each operation takes constant time.

### 5.2.2 Symmetry-Type Conversion

There are six cases to convert a symmetry type, and each of them takes linear time.

- Case 1: 1D vertical to horizontal symmetry.
- Case 2: 1D horizontal to vertical symmetry.
- Case 3: 1D vertical symmetry to 2D symmetry.
- Case 4: 1D horizontal symmetry to 2D symmetry.
- Case 5: 2D symmetry to 1D vertical symmetry.
- Case 6: 2D symmetry to 1D horizontal symmetry.

In Cases 1 and 2, to convert the symmetric placement between two 1D symmetry types, we first rotate each module, and then swap the left and the right children of each node.

In Cases 3 and 4, we first need to check if there is only one self-symmetry module such that the symmetric-feasible condition is satisfied. To convert the symmetric placement from 1D symmetry to 2D symmetry, the representative of each symmetry pair and self-symmetry module should be redetermined and the ASF-B*-tree is rebuilt.

In Cases 5 and 6, to convert the symmetric placement from 2D symmetry to 1D symmetry type, we keep the tree structure and update each representative for the converted symmetry type.

## 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. 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.

Figure 9 shows an example of updating contour related nodes. Initially, there are five contour nodes in the HB*-tree. After perturbing the ASF-B*-tree of the symmetry group $S_0$, the contour nodes $n_{03}$ and $n_{04}$ are disappeared, and the nodes $n_5$ and $n_8$ become dangling nodes. We first find the nearest contour node of $n_5$, which is $n_{02}$. Since $n_{02}$ has no right child, we simply assign $n_5$ to be its right child and $n_{02}$ is the parent of $n_5$. Then, the nearest contour node of $n_8$ is searched, which is also $n_{02}$. Since $n_{02}$ already has the right child $n_5$, the most left-skewed child is searched, which is $n_6$. We assign $n_6$ to be the parent of $n_8$, and $n_8$ is the left child of $n_6$.

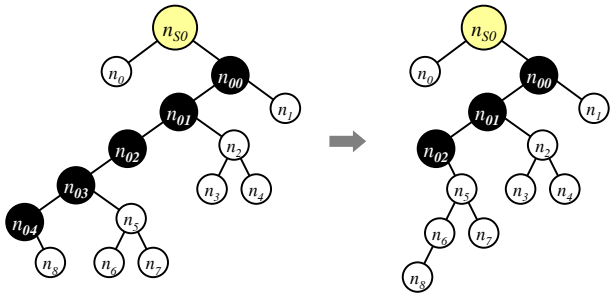



**Figure 9: An example of updating contour related nodes.**

We have the following theorem for the packing complexity.

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

It should be noted that this is the fastest algorithm in the literature for the placement with symmetry constraints. Previous works take $\Omega(n \lg n)$ times; for example, both of the most recent works [9, 14] take $O(n^2)$ time for the packing.

# 6. EXPERIMENTAL RESULTS

We implemented our placement algorithm in the C++ programming language on a 3.2GHz Intel Pentium4 PC under the Linux operation system. We performed two sets of experiments: one is based on the four MCNC benchmarks (apte, hp, ami33, and ami49) used in [11], and the other is two real industry analog designs (biasynth_2p4g and lnamixbias_2p4g) used in [5] and [9]. (Note that the work [5] tested on six industry designs, but only two of them are available to the public [9].)

In the first set of experiments, we compared our algorithm with the following works: sequence pairs [1], segment trees [3], TCG-S [11], and sequence pairs with dummy nodes [14]. Table 1 lists the names of the MCNC benchmark circuits ("Circuit"), the numbers of modules ("# of Mod."), the numbers of symmetry modules ("# of Sym. Mod."), the total module areas ("Mod. Area"), the total areas ("Area") and the runtimes ("Time") for the aforementioned works and our HB*-tree with area optimization alone,

**Table 1: 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 | # of Mod. | # of Sym. Mod. | Mod. Area ($mm^2$) | SP [1] | | Seg. Tree [3] | | TCG-S [11] | | SP w. Dummy [14] | | HB*-tree | | HB*-tree (Area + WL) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Area | Time (s) | Area | Time (s) | Area | Time (s) | Area | Time (s) | Area | Time (s) | Area | HPWL ($mm$) | Time (s) |
| apte | 9 | 8 | 46.56 | 48.12 | 25 | 47.52 | 11 | 47.52 | 3 | 46.92 | 13 | 46.92 | 2 | 47.90 | 10.20 | 3 |
| hp | 11 | 8 | 8.83 | 9.84 | 138 | 9.71 | 62 | 9.71 | 50 | 9.43 | 13 | 9.35 | 2 | 10.10 | 30.74 | 16 |
| ami33 | 33 | 6 | 1.16 | 1.24 | 684 | 1.23 | 307 | 1.21 | 423 | 1.24 | 23 | 1.23 | 12 | 1.29 | 47.23 | 39 |
| ami49 | 49 | 4 | 35.45 | 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 2: Comparisons of area utilization and CPU times for sequence pair (SP) (on Sun Blade 100 500MHz), segment tree (Seg. Tree) (on Sun Blade 100 500MHz), SP+LP (Pentium4 3.2GHz), sequence pair with dummy nodes (SP w. Dummy) (on Pentium4 3.2GHz), and HB*-tree (on Pentium4 3.2GHz), based on two real industry benchmarks.**

| Circuit | # of Mod. | # of Sym. Mod. | Mod. Area ($10^3 \mu m^2$) | SP [1] | | Seg. Tree [5] | | SP+LP [9] | | SP w. Dummy [14] | | HB*-tree | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Area | Time (s) | Area | Time (s) | Area | Time (s) | Area | Time (s) | Area | Time (s) |
| biasynth_2p4g | 65 | 8+12+5 | 4.70 | 5.40 | 780 | 5.40 | 246 | 5.00 | 403 | 5.57 | 134 | 4.92 | 22 |
| lnamixbias_2p4g | 110 | 16+6+6+12+4 | 46.00 | 50.80 | 2824 | 50.30 | 726 | 49.95 | 3252 | 52.21 | 227 | 48.63 | 43 |
| Comparison | | | | 1.05 | - | 1.04 | - | 1.03 | 46.96 | 1.08 | 5.68 | 1.00 | 1.00 |

same as the previous works, and with simultaneous area and wirelength optimization. The results of the works [1, 3, 11] are taken from the paper [11], and those of [14] 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 [1], [3], [11], and [14], respectively. Noted that the improvements should not be considered marginal since the previous works have pushed the solution quality close to their limits. For the running time, our algorithm is about 4.09X faster than [14]. 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 sequence pairs in [1], segment trees in [5], sequence pairs with linear programming in [9], and sequence pairs with dummy nodes in [14]. The results show that our algorithm achieved average area reductions of 5%, 4%, 3%, and 8% over [1], [5], [9], and [14], respectively. For the running time, our algorithm is about 46.96X and 5.68X faster than those in [9] and [14], respectively. Again, the previous works [1, 5] 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. Figure 10 shows the resulting placement of biasynth_2p4g, with the symmetry modules being colored.

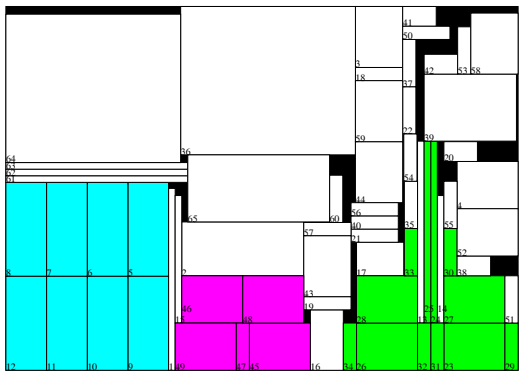


**Figure 10: The resulting placement of biasynth_2p4g with three symmetry groups in 1D vertical symmetric placement.**

# 7. CONCLUSIONS

We have presented the first amortized linear-time packing algorithm for the placement with symmetry constraints. We have introduced the concept of symmetry islands and presented the ASF-B*-trees to directly model the placement of a symmetry island. Our algorithm is the first work to consider both 1D and 2D symmetry constraints. We have also presented the hierarchical HB*-trees to simultaneously optimize the placement with both symmetry islands and non-symmetry modules. Experimental results have shown that our approach achieves the best published quality and runtime efficiency for analog placement.

# 8. ACKNOWLEDGMENTS

We would like to thank Prof. Yao-Wen Chang of National Taiwan University for supervising this work and revising this paper. We also would like to thank Prof. Evangeline F. Y. Young and Mr. Yiu-Cheong Tam of the Chinese University of Hong Kong for providing the package of their work [14] for the comparative studies. This work was partially supported by Springsoft, Inc. and National Science Council of Taiwan under Grant No's NSC 95-2221-E-002-372, NSC 95-2221-E-002-374, and NSC 95-2752-E-002-008-PAE.

# 9. REFERENCES

[1] F. Balasa and K. Lampaert, "Symmetry within the sequence-pair representation in the context of placement for analog design," *IEEE TCAD*, vol. 19, no. 3, pp. 721–731, Jul. 2000.

[2] F. Balasa, "Modeling non-slicing floorplans with binary trees," *Proc. ICCAD*, pp. 13–16, 2000.

[3] F. Balasa , S. C. Maruvada, and K. Krishnamoorthy, "Efficient solution space exploration based on segment trees in analog placement with symmetry constraints," *Proc. ICCAD*, pp. 497–502, 2002.

[4] F. Balasa , S. C. Maruvada, and K. Krishnamoorthy, "Using red-black interval trees in device-level analog placement with symmetry constraints," *Proc. ASPDAC*, pp. 777–782, 2003.

[5] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy, "On the exploration of the solution space in analog placement with symmetry constraints," *IEEE TCAD*, vol. 23, no. 2, pp. 177–191, Feb. 2004.

[6] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-Trees: a new representation for non-slicing floorplans," *Proc. DAC*, pp. 458–463, 2000.

[7] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Charley, "KOAN/ANAGRAM II: New tools for device-level analog placement and routing," *IEEE JSSC*, vol. 26, pp. 330–342, Mar. 1991.

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

[9] S. Kouda, C. Kodama, and K. Fujiyoshi, "Improved method of cell placement with symmetry constraints for analog IC layout design," *Proc. ISPD*, pp. 192–199, 2006.

[10] J.-M. Lin, H.-E. Yi, and Y.-W. Chang, "Module placement with boundary constraints using B*-trees," *IEE Proceedings – Circuits, Devices and Systems*, vol. 149, no. 4, pp. 251–256, Aug. 2002.

[11] J.-M. Lin, G.-M. Wu, Y.-W. Chang, and J.-H. Chuang, "Placement with symmetry constraints for analog layout design using TCG-S," *Proc. ASPDAC*, pp. 1135–1138, 2005.

[12] S. C. Maruvada, A. Berkman, K. Krishnamoorthy, and F. Balasa , "Deterministic skip lists in analog topological placement," *Proc. ASICON*, pp. 756–759, Oct. 2005.

[13] Y.-X. Pang, F. Balasa , K. Lampaert, and C.-K. Cheng, "Block placement with symmetry constraints based on the O-tree non-slicing representation," *Proc. DAC*, pp. 464-467, 2000.

[14] Y.-C. Tam, E. F.-Y. Young and C. Chu, "Analog placement with symmetry and other placement constraints," *Proc. ICCAD*, pp. 349–354 2006.

[15] G.-M. Wu, Y.-C. Chang, and Y.-W. Chang, "Rectilinear block placement using B*-trees," *ACM Trans. on Design Automation of Electronic Systems*, vol. 8., no. 2, pp. 188–202, Apr. 2003.