Floorplan Representations: Complexity and Connections

BO YAO, HONGYU CHEN, CHUNG-KUAN CHENG, and RONALD GRAHAM
University of California, San Diego

Floorplan representation is a fundamental issue in designing a floorplanning algorithm. In this paper, we first present a twin binary trees structure for mosaic floorplans. It is a nonredundant representation. We then derive the exact number of configurations for mosaic floorplans and slicing floorplans. Finally, the relationships between various state-of-the-art floorplan representations are discussed and explored.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids—*Layout*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Floorplan representation, mosaic floorplan, twin binary trees, O-tree, number of combinations, Baxter permutation

1. INTRODUCTION

Floorplanning is becoming more and more important in VLSI physical design because circuit size is growing rapidly and hierarchical design with IP blocks is now widely used to reduce the design complexity. Unfortunately, many floorplanning problems are NP-complete, so most floorplanning algorithms use either analytical force-directed methods or perturbations with random searches and heuristics. Floorplan representation thus becomes a fundamental issue since the efficiency of these basic operations relies on the geometrical expression of circuit blocks.

There are two important characteristics of an efficient floorplan representation. The first is the number of the combinations of a representation. A combination of a representation is a distinct coding in that representation. It is possible that two distinct combinations of a representation method actually correspond to the same floorplan. We refer to this case as the *redundancy* of the representation. A method that can represent more general floorplans and has less redundancy is desirable. The second aspect is the time complexity of the transformation between a representation and its corresponding floorplan.

This work was supported in part under grants from NSF project MIP-9987678 and the California MICRO program.
Authors' addresses: CSE Department, University of California, San Diego, La Jolla, CA 92093-0114; email: {byao,hchen,kuan,rgraham}@cs.ucsd.edu.

Since this transformation is the most frequently used operation of a floorplanner, we want its time complexity to be as low as possible.

1.1 State-of-The-Art in Floorplan Representations

The representation of a floorplan has been intensively studied over the past few decades. Most of these representations have addressed the completeness of a floorplan topology. For a floorplan with a slicing structure [Otten 1982], the v-h tree was one of the earliest representations [Szepieniec and Otten 1980]. A binary tree representation was also widely used. The leaves of the binary tree correspond to the blocks and each internal node defines a vertical or horizontal merge operation of the two descendents. It is known that an upper bound on the number of possible configurations for this binary tree representation is given by the expression $O(n!2^{5n-3}/n^{1.5})$ [Knuth 1997], where n is the number of blocks in the floorplan.

For a more general nonslicing floorplan, the earliest representation is the constraint graph, for example, the Grason graph [Otten 2000; Grason 1970] and the polar graph [Grason 1970; Kozminsky and Kinnen 1985]. Later, in the mid-1990s, two efficient representations, sequence pair [Murata et al. 1995] and the bounded-sliceline grid [Nakatake et al. 1998], were proposed.

Murata et al. [1995] proposed a sequence pair (SP) representation. They used two sets of permutations to represent the topological relations between blocks. Thus the number of combinations of the SPs is $O((n!)^2)$. It takes $O(n \log\log n)$ time to transform between an SP and its corresponding block level placement [Tang and Wong 2001].

Nakateke et al. [1998] introduced a bounded-sliceline grid (BSG) approach. A special n-by-n grid was devised for placing n blocks. This approach has $n!\binom{n^2}{n}$combinations and contains a lot of redundancy. The time complexity of the transformation is $O(n^2)$.

More recently, Hong et al. [2000] proposed a corner block list (CB) representation of a mosaic floorplan. Mosaic floorplans cover all slicing floorplans and all nonslicing floorplans with n rectangles for n blocks. The concept of a mosaic floorplan was also previously referred to as a floorplan with no empty rooms. Sakanushi and Kajitani [2000] also proposed an equivalent representation for mosaic floorplans named the quarter-state sequence (Q-sequence). The CB representation has a relatively small combination number $O(n!2^{3n-3}/n^{1.5})$ compared to the SP or the BSG, and the time complexity of the transformation is $O(n)$.

Guo et al. [1999] proposed an ordered tree (O-tree) representation of a nonslicing floorplan. An O-tree represents partial topological information, which together with the dimensions of all the blocks describes an exact floorplan. The O-tree has a combination number $O(n!2^{2n-2}/n^{1.5})$, which is smaller than that for CBs. The approach produces a floorplan in time $O(n)$.

1.2 Our Contributions

In this paper, we find the exact numbers of slicing and mosaic floorplans. An efficient representation of nonslicing floorplans is developed. We also study the

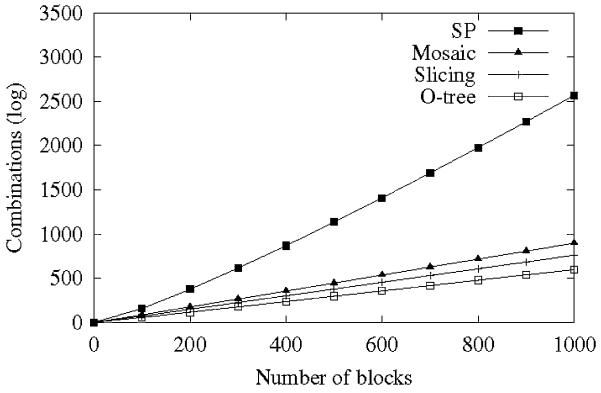


Fig. 1. Combinations of floorplans and representations.

Table I. Exact Number of Combinations of Different Floorplan Configurations and Representations

Number of blocks	Combinations of O-tree	Combinations of slicing floorplan	Combinations of mosaic floorplan	Combinations of sequence pairs
1	1	1	1	1
2	2	2	2	2
3	5	6	6	6
4	14	22	22	24
5	42	90	92	120
6	132	394	422	720
7	429	1,806	2,074	5,040
8	1,430	8,558	10,754	40,320
9	4,862	41,586	58,202	362,880
10	16,796	206,098	326,240	3,628,800
11	58,786	1,037,718	1,882,690	39,916,800
12	208,012	5,293,446	11,140,560	479,001,600
13	742,900	27,297,738	67,329,992	6,227,020,800
14	2,674,440	142,078,746	414,499,438	87,178,291,200
15	9,694,845	745,387,038	2,593,341,586	1,307,674,368,000
16	35,357,670	3,937,603,038	16,458,756,586	20,922,789,888,000
17	129,644,790	20,927,156,706	105,791,986,682	355,687,428,096,000

relationships between some of these representations. Our contributions in this paper include the following results:

(1) We find the exact numbers of two kinds of floorplans: mosaic floorplans and slicing floorplans. The number of mosaic floorplans is a Baxter number [Baxter 1964; Chung et al. 1978], and that of slicing floorplans is a Schröder number [Etherington 1940]. The combination numbers of SPs, mosaic floorplans, slicing floorplans, and O-trees are illustrated on a log scale in Figure 1. The combination numbers are normalized by $n!$, which is the number of permutations of n blocks. The slopes of the lines for mosaic floorplans, slicing floorplans, and O-tree structures are the constants 0.89, 0.76, and 0.59, respectively. On the other hand, the slope of the line for sequence pair increases with a rate of $\log n$. Table I provides the

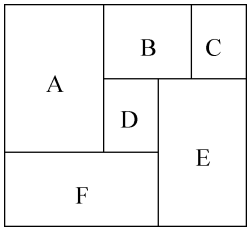


Fig. 2. An example of a mosaic floorplan.

exact numbers of the combinations for a number of blocks ranging from 1 to 17.

(2) We present a new representation, a "twin binary trees" structure, for mosaic floorplans. We also revisit the v-h tree representation for slicing floorplans. These two structures have zero redundancy because they give an exact one-to-one mapping to their corresponding floorplans.

(3) We study the relationships between SPs, O-Trees, CBs, twin binary trees, and v-h trees. This study points out the redundancy of different representations.

2. THE NUMBER OF FLOORPLANS AND REPRESENTATIONS WITH A ONE-TO-ONE MAPPING

In this section, we describe the one-to-one mapping representations for mosaic and slicing floorplans, respectively. We then derive the exact numbers of the two different kinds of floorplans.

2.1 Mosaic Floorplans and Twin Binary Trees Representations

Mosaic floorplans were introduced in Hong et al. [2000]. Mosaic floorplans have the following characteristics:

(1) There is no empty space within the floorplan, that is, each rectangle is assigned to one and only one block. In the floorplan area, except the four corners of the chip, the segment intersection forms a "T-junction." A T-junction is composed of a noncrossing segment and a crossing segment. The noncrossing segment has one end touching the crossing segment.

(2) The topology is equivalent before and after the noncrossing segment of the T-junction slides to adjust the block sizes.

(3) There is no degenerate case where two distinct T-junctions meet at the same point.

Figure 2 illustrates a mosaic floorplan example. In general, we can summarize the above by noting that the set of slicing floorplans is a subset of the set of mosaic floorplans, which in turn is a subset of the set of general floorplans (Figure 3).

2.1.1 *Encoding Mosaic Floorplans with Twin Binary Trees.* We now propose a tree structure representation that captures the topological relationships

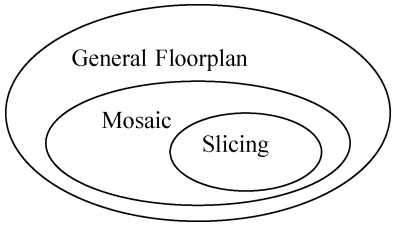
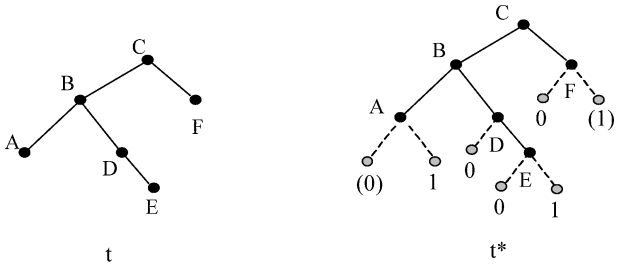


Fig. 3. Categories of floorplans.



$\Theta(t) = 10010 \quad I(t) = ABDECF$

Fig. 4. An example of an augmented binary tree.

of the blocks. Given a mosaic floorplan with n blocks, our representation consists of a pair of binary trees. Each binary tree has n nodes, and each node corresponds to one block in the floorplan.

Definition 2.1.1 (*Augmented Binary Tree*). Given a binary tree t, the augmented binary tree t^* is a tree generated by the following augmentation: To each node in tree t, we add a left child node with label 0 if the node has no left child, and a right child node with label 1 if it has no right child.

Figure 4 gives an example of an augmented binary tree. A binary tree t with six nodes is drawn on the left. The augmented binary tree t^* of the binary tree t is drawn on the right. The augmented nodes of the tree t^* are drawn in a light color and the augmented edges are shown as dashed lines.

Definition 2.1.2 (*The Order I*(t) *and the* Θ*-Labeling of a Binary Tree* t). Given a binary tree t and its augmented binary tree t^*, we carry out in-order traversals on them. The traversal visits the nodes in a recursive order, left branch first, root second, and right branch last. The order of a binary tree t, I(t), is the node sequence of the tree t from the in-order traversal. The Θ-labeling, $\Theta(t)$, is the leaf label sequence of tree t^* ignoring the first and the last bits.

For example, in Figure 4, t is a binary tree with 6 nodes and t^* is its corresponding augmented binary tree. The in-order traversal visits the augmented binary tree in the sequence 0A1B0D0E1C0F1. The order of the binary tree is $I(t) = \text{ABDECF}$, and the Θ-labeling is $\Theta(t) = 10010$.

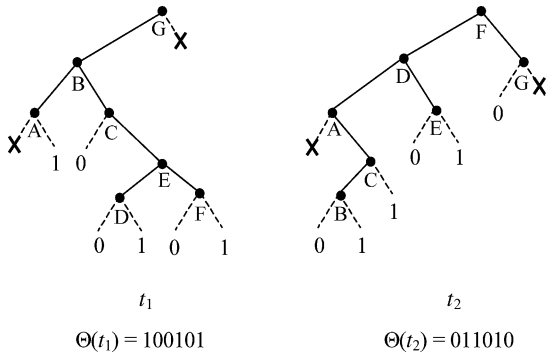


Fig. 5. An example of twin binary trees.

The Properties of the Θ-*labeling* are as follows:

(1) The first bit encountered is always 0 and the last bit encountered is always 1. Therefore we ignore these two bits in the Θ-labeling of a binary tree (Figure 4).
(2) The length of $\Theta(t)$ is equal to $n - 1$ if the binary tree t has n nodes. For example, the binary tree t in Figure 4 has six nodes and the length of labeling $\Theta(t)$ is 5.
(3) The in-order traversal on the augmented binary tree t^* visits the leaf nodes and the internal nodes alternately. Therefore when the pth internal node is visited, the number of traversed bits in $\Theta(t)$ is $p - 1$. For example in Figure 4, node C is the fifth node traversed. The number of bits in $\Theta(t)$ traversed before node C is 4 (1001).

We denote the bit-wise complement of $\Theta(t)$ by $\Theta^c(t)$, that is, the two strings have equal length, $|\Theta^c(t)| = |\Theta(t)|$, and each bit b in $\Theta(t)$ is replaced by $1 - b$ in $\Theta^c(t)$. For example if $\Theta(t) = 10010$, then $\Theta^c(t) = 01101$.

Definition 2.1.3 (*Twin Binary Trees*). Let Tree_n be the set of binary trees with n nodes. The set of *twin binary trees* TBT_n is defined as follows: $\text{TBT}_n = \{(t_1, t_2) | t_1, t_2 \in \text{Tree}_n, \text{I}(t_1) = \text{I}(t_2), \text{and } \Theta(t_1) = \Theta^c(t_2)\}$.

Figure 5 shows an example of twin binary trees with $n = 7$. Each tree, t_1 and t_2, has seven nodes. The orders of these two trees are $I(t_1) = I(t_2) = \text{ABCDEFG}$. The Θ-labeling $\Theta(t_1)$ is 100101 and $\Theta(t_2)$ is 011010. Thus, $\Theta(t_1) = \Theta^c(t_2)$. Note that the first and the last labels are not included in the Θ-labelings. In Figure 5, we denote both of these two labels by a cross.

Definition 2.1.4 (*0° T-Junctions,* 90°*T-Junctions,* 180°*T-Junctions, and* 270°*T-Junctions*). In a mosaic floorplan, except for the four corners of the boundary rectangle, a T-shaped intersection has four different orientations (Figure 6). We call these four kinds of intersections 0°*T-junctions,* 90°*T-junctions,* 180°*T-junctions*, and 270°*T-junctions*, respectively.

Definition 2.1.5 (*C^+-Parent and C^+-Child*). Given a mosaic floorplan, assume that block A is not the up-right corner block of the floorplan. The T-junction

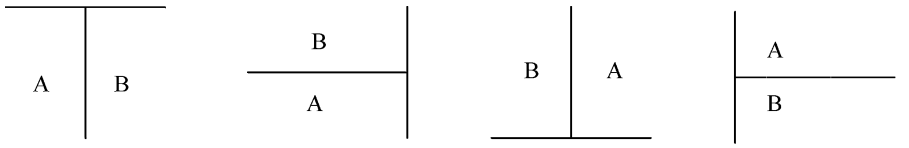


Fig. 6. Four different types of T-junctions.

at the up-right corner of A is either a 0° T-junction or a 270° T-junction. Let B be the block adjacent to A by the noncrossing segment at the corner of that T-junction. B is called the C^+-*parent* of A and A is called the C^+-*child* of B (Figures 6(a) and 6(b)).

Definition 2.1.6 (C^--*Parent and* C^--*Child*). Given a mosaic floorplan, assume that block A is not the bottom-left corner block of the floorplan. The T-junction at the bottom-left corner of A is either a 90° T-junction or a 180° T-junction. Let B be the block adjacent to A by the noncrossing segment at the corner of that T-junction. B is called the C^--*parent* of A and A is called the C^--*child* of B (Figures 6(c) and 6(d)).

The following three properties describe the tree properties of the C^+- and C^--relations.

PROPERTY 2.1.7. *Except for the upper-right corner block of the floorplan, each block of the floorplan has exactly one* C^+*-parent. Except for the bottom-left corner block of the floorplan, each block of the floorplan has exactly one* C^-*-parent.*

PROPERTY 2.1.8. *Each block of the floorplan has at most one* C^+*-child adjacent to its left edge and at most one* C^+*-child adjacent to its bottom edge.*

PROPERTY 2.1.9. *Each block of the floorplan has at most one* C^-*-child adjacent to its right edge and at most one* C^-*-child adjacent to its top edge.*

The algorithm converting a mosaic floorplan to twin binary trees (**MFTB**) is shown in Figure 7. Given a mosaic floorplan, we denote each block with a node. For each block i and its C^+-parent j, we have node j branch to node i. Depending on the direction of the T-junction, if the C^+-parent j is on the right-hand side of the block i, we set node i to be the left child of node j. Otherwise, node i is set to be the right child of node j. The result is a tree, τ_+, rooted at the upper-right corner block of the floorplan. Similarly, we connect each node to its C^--parent and get a tree τ_- rooted at the bottom-left corner block.

LEMMA 2.1.10. *Algorithm MFTB produces a unique pair of binary trees* τ_+ *and* τ_-.

PROOF. Both the trees τ_+ and τ_- are binary trees because of Properties 2.1.8 and 2.1.9. The resulting tree pair produced by the algorithm MFTB is unique because of Properties 2.1.7 to 2.1.9. Because for each block, there is at most one C^+-child adjacent to its bottom edge, and at most one C^+-child adjacent to its left edge, the algorithm MFTB builds a unique τ_+. Similarly, algorithm MFTB

Algorithm MFTB

Input: A mosaic floorplan with n blocks.

Output: The tree pair (τ_+, τ_-).

1. Initialize two sets of nodes V^+, V^- and two sets of edges E^+, E^- to empty sets.
2. Let $V^+ = V^- = \{i \mid$ there is a block labeled i in the floorplan$\}$
3. For each block i do
4. If i is not the up-right corner block of the floorplan, then
5. Get C^+-parent j of block i and add (j, i) to edge set E^+
6. If block j is on the right of block i, let i be the left child of j
7. Else let i be the right child of j.
8. If i is not the bottom-left corner block of the floorplan, then
9. Get C^--parent j of block i and add (j, i) to edge set E^-
10. If block j is on the left of block i, let i be the right child of j,
11. Else let i be the left child of j.
12. Endfor
13. Let $\tau_+ = (V^+, E^+)$, $\tau_- = (V^-, E^-)$.
14. Return (τ_+, τ_-).

Fig. 7. The algorithm to produce the twin binary tree representation of a mosaic floorplan.

builds a unique τ_- for the mosaic floorplan. Therefore, the algorithm MFTB outputs a unique pair of binary trees. □

Figure 8 shows an example of the twin binary trees representation of a mosaic floorplan. In Figure 8(a), the mosaic floorplan in Figure 2 is redrawn. In Figure 8(b), we present the binary tree representation, (τ_+, τ_-), for the floorplan in Figure 8(a). To make the derivation of the twin binary trees clear, in Figure 8(a), we mark the upper-right corner of each block with a circle. These circles correspond to the nodes in τ_+. We mark the lower-left corner of each block with a cross. These crosses correspond to the nodes in τ_-. The line segments between adjacent circles are marked with bold solid lines, which correspond to the edges in τ_+. The line segments between adjacent crosses are marked with bold dashed lines, which correspond to the edges in τ_-.

In Theorem 2.1.11, we shall prove that the resulting binary tree pair, (τ_+, τ_-), is a *twin binary trees representation*.

THEOREM 2.1.11. *The MFTB algorithm produces a pair of binary trees* (τ_+, τ_-) *satisfying two properties:*

- *Order of the Trees:* $I(\tau_+) = I(\tau_-)$.
- Θ*-labeling:* $\Theta(\tau_+) = \Theta^C(\tau_-)$.

PROOF. See Appendix 1. □

For the time complexity of algorithm MFTB, we have the following theorem.

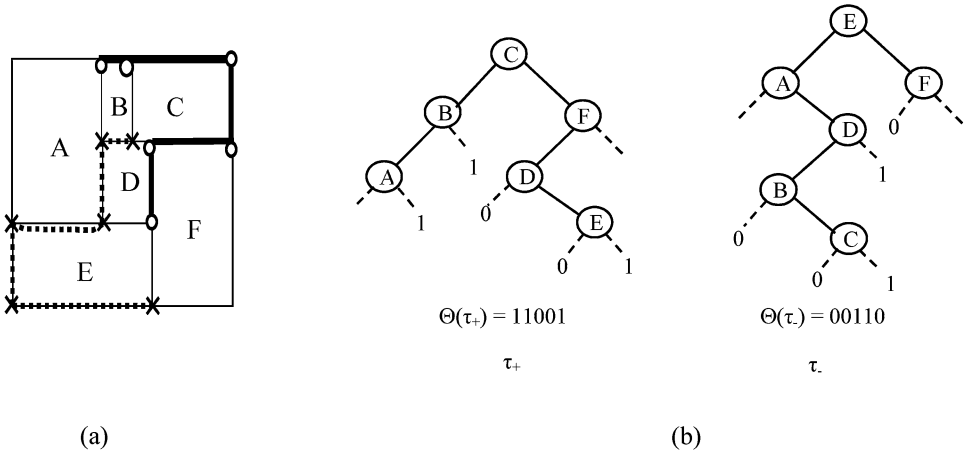


Fig. 8. A twin binary trees representation of a mosaic floorplan.

THEOREM 2.1.12. *The time complexity of algorithm MFTB is $O(n)$, where n is the number of blocks in the floorplan.*

PROOF. Algorithm MFTB scans through all the n blocks in the floorplan once. For each block, it checks the T-junction at the upper-right end, finds its C^+-neighbor and adds an edge in τ_+. This can be done in $O(1)$ time. It also checks the T-junction at the lower-left corner, finds its C^--neighbor and adds an edge in τ_-. This can also be done in $O(1)$ time. Therefore, the time complexity of algorithm MFTB is $O(n)$. □

2.1.2 *Decoding Twin Binary Trees into a Mosaic Floorplan.* Given a pair of twin binary trees, we can find the corresponding mosaic floorplan for it. (Figure 9). The floorplan is described with a horizontal constraint graph G_H and a vertical constraint graph G_V. Each vertex of the constraint graph represents a line segment in the floorplan and each edge between vertices represents a block (Figure 9(b)).

We first introduce the relations of the leaf nodes on the augmented twin binary trees and the line segments in the mosaic floorplan. Then we present the algorithm TBMF to decode the twin binary trees into a mosaic floorplan.

Figure 9(a) shows an example of augmented twin binary trees ($\tau_+{}^*$,$\tau_-{}^*$). The original twin binary trees are drawn with character-labeled nodes and solid line edges. The leaves are now marked with numbers. Likewise, we also mark with numbers the vertices of the constraint graphs G_H and G_V (Figure 9(b)). We have the following properties regarding the leaf nodes on the augmented twin binary trees:

(1) The leaves of the augmented twin binary trees ($\tau_+{}^*$,$\tau_-{}^*$) correspond to the line segments in the floorplan. The type of the leaf node represents the direction of the line segment. If the leaf is a left child in $\tau_+{}^*$ or a right child in $\tau_-{}^*$, it represents a horizontal line segment. If the leaf is a right child in $\tau_+{}^*$ or a left child in $\tau_-{}^*$, it represents a vertical line segment. For example, the leaf

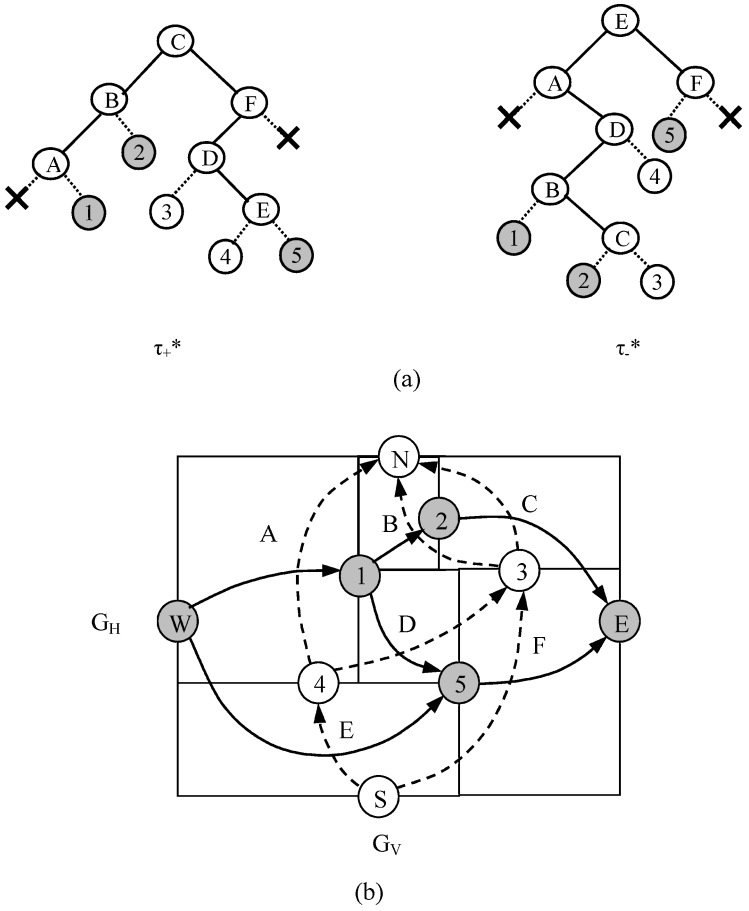


Fig. 9. The twin binary trees representation of a mosaic floorplan.

marked 3 in $\tau_+{}^*$ of Figure 9(a) represents a horizontal line segment (Figure 9(b)), and the leaf marked 5 in $\tau_-{}^*$ represents a vertical line segment.

(2) The kth leaves of $\tau_+{}^*$ and $\tau_-{}^*$ represent the kth line segments in the corresponding floorplan.

Definition 2.1.13 (*The Directed Ancestor Set*). Given a node k in the binary tree t, the *directed ancestor set* of k, D(k, t), is defined as follows:

D(k, t) = {v|v is a node in the tree t and node k can be reached by consecutive left branches from node v if k is a left child, or node k can be reached by consecutive right branches from node v if k is a right child}.

For example, in the binary tree $\tau_+{}^*$ of Figure 9(a), leaf 3 is a left child of node D, and leaf 3 can be reached by two consecutive left branches from node F. Thus, the directed ancestor set of leaf 3 in $\tau_+{}^*$, D(3, $\tau_+{}^*$) = {D, F}.

By using either consecutive left branches or right branches, a leaf can always be reached from some internal node. Hence, each internal node e of the binary tree t is in exactly two different directed ancestor sets of the leaf node. Based

on this property, we have procedure FindD(t) to find the directed ancestor sets of all leaf nodes in the binary tree t.

Procedure FindD(t)
Precondition: t is the root of an augmented binary tree. For all the leaf nodes k, the global variables D(k, t) are initialized to empty sets.
/* Use $t.left$ and $t.right$ to denote the left and right child, respectively.$t.lda$ and $t.rda$ represent the leaf node which can be reached by consecutive left branches and consecutive right branches, respectively. */

```
1.  If t.left is a leaf node in t
2.    Then t.lda = t.left
3.    Else call procedure FindD(t.left)
4.         t.lda = t.left.ldat
5.  If t.right is a leaf node in t
6.    Then t.rda = t.right
7.    Else call procedure FindD(t.right)
8.         t.rda = t.right.rda
9.  Add node t to the sets D(t.lda, t) and D(t.rda, t)
10. Return
```

The following is the property of the directed ancestor set of a leaf node:

> The directed ancestor set of a leaf of the augmented twin binary trees represents the blocks on each side of a line segment. If leaf k is a left child in the augmented binary tree $\tau_+{}^*$, leaf k corresponds to the horizontal segment k in a floorplan, and the directed ancestor set D(k, $\tau_+{}^*$) represents the blocks lying just below the line segment k. Similarly, if leaf k is a right child in $\tau_+{}^*$, then D(k, $\tau_+{}^*$) represents the blocks lying just right of the line segment k.

For example, in the binary tree τ_+ of Figure 9(a), the directed ancestor set of the leaf node 3 is the set {D, F}, so the two blocks, D and F lie right below the horizontal line segment 3 of the floorplan.

The algorithm TBMF for decoding twin binary trees to a floorplan is listed in Figure 10. The input to the algorithm is a pair of twin binary trees, (τ_+, τ_-), with n nodes each. The output is the constraint graphs of the corresponding mosaic floorplan. We first construct the augmented twin binary trees, $\tau_+{}^*$ and $\tau_-{}^*$ by Definition 2.1.1 and find the directed ancestor set of each leaf node in the augmented twin binary trees by Definition 2.1.13.

To generate the constraint graphs of the corresponding mosaic floorplan, we traverse in augmented binary trees $\tau_+{}^*$ and $\tau_-{}^*$ in order (Definition 2.1.2). Let node e_+ be the kth leaf traversed in tree $\tau_+{}^*$ and node e_- be the kth leaf traversed in tree $\tau_-{}^*$. If this node pair represents a vertical line segment, we add a new node v to the horizontal constraint graph G_H. For each element in the directed ancestor set D(e_-, $\tau_-{}^*$), we create a new link leaving vertex v in G_H. Also, for each element in set D(e_+, $\tau_+{}^*$), we make the corresponding link e in G_H go into the node v. Similarly, if the leaf node pair represents a horizontal line segment, we add a new vertex v to the vertical constraint graph G_V. For each element in the directed ancestor set of D(e_+, $\tau_+{}^*$), we create a new link going into vertex v in G_V. Also, for each element in the set D(e_-, $\tau_-{}^*$), we make the corresponding link in G_V go out from v.

Algorithm TBMF
Input: Twin binary trees (τ_+ , τ_-) of n nodes each.
Output: Constraint graphs G_H and G_V.

1. Given a name sequence { A, B, C, … }, label the nodes of the two trees in-order (Definition 2.1.2).
2. Get the augmented twin binary trees (τ_+^*,τ_-^*).
3. Initialize G_H and G_V to empty graphs.
4. Call procedure FindD(τ_+^*) and FindD(τ_-^*)
5. Do in-order traversal on τ_+^* and τ_-^*
5. For the k-th leaf e_+ in τ_+ and the k-th leaf e_- in τ_- do
6. Case k = 1 /* The first pair */
7. Create source node S in G_H.
8. For each element in D(e_-, τ_-^*), create an outgoing edge from S.
9. Create sink node T in G_V.
10. For each element in D(e_+, τ_+^*), create an incoming edge to T.
11. Case k = n+1 /* The last pair */
12. Create sink node T in G_H.
13. Collect all un-connected outgoing edges in G_H and make them point to T.
14. Create source node S in G_V.
15. Collect all non-connected incoming edges in G_V and direct them out from S.
16. Otherwise
17. If e_+ has label 0
18. Create a node v in G_V.
19. For each element in D(e_+, τ_+^*), create an incoming edge to v in G_V.
20. For each element in D(e_-, τ_-^*), collect the corresponding edges in G_V and direct them out from v.
21. Else
22. Create a node v in G_H.
23. For each element in D(e_-, τ_-^*), create an outgoing edge from v in G_H.
24. For each element in D(e_+, τ_+^*), collect the corresponding edges in G_H and direct them into v.
25. Endfor

Fig. 10. The algorithm to construct a mosaic floorplan from a pair of twin binary trees.

For example, the second leaf node (labeled 1) of tree τ_+^* in Figure 9(a) is the right child of node A. Its directed ancestor set is {A}. The second leaf node (labeled 1) of τ_-^* is the left child of node B. Its directed ancestor set is {B, D}. This leaf pair corresponds to a vertical line segment. Thus, we add a vertex v in G_H, direct the link A to vertex v, add links B and D and direct them from vertex v.

Note that the first pair and the last pair of leaves in the augmented binary trees are two special cases. The first leaf e_+ in tree τ_+^* represents the top boundary line of the floorplan. We add the sink node T to the vertical constraint graph G_V. The first leaf e_- in tree τ_-^* represents the left boundary line of the floorplan, so we add the source node S to the horizontal constraint graph G_H. The operation on the last pair is similar.

Figure 9(b) draws the resulting constraint graph for the floorplan defined by the twin binary trees in Figure 9(a). The horizontal constraint graph G_H is

drawn with dark colored nodes and solid line edges, while the vertical constraint graph G_V is drawn with light colored nodes and dash line edges.

THEOREM 2.1.14. *The time complexity for algorithm TBMF is $O(n)$, where n is the number of nodes in each twin binary tree.*

PROOF. The first three steps of TBMF can all be done in time $O(n)$. The fourth step of TBMF also runs in $O(n)$ time, because the procedure FindD visits each node of the tree exactly once. The remaining steps add the edges to the constraint graph. The total number of edges in G_H and G_V is $2n$, because each block corresponds to one edge in G_H and one in G_V. Each edge can be added in time $O(1)$. Therefore, the total running time of algorithm TBMF is $O(n)$. □

THEOREM 2.1.15. *There is a one-to-one mapping between a pair of twin binary trees with n nodes and a mosaic floorplan with n blocks.*

PROOF. Given a mosaic floorplan, algorithm MFTB outputs a unique pair of twin binary trees corresponding to the floorplan. This is true because of Lemma 2.1.10 and Theorem 2.1.11. Conversely, given a pair of twin binary trees, algorithm TBMF finds a unique mosaic floorplan corresponding to it. Because of the property of the directed ancestor set of a leaf node in an augmented tree, we know that the directed ancestor set of a leaf node represents the blocks in each side of a line segment. □

2.1.3 *The Exact Number of Mosaic Floorplans.* The preceding two sections established a one-to-one correspondence between twin binary trees and a mosaic floorplan, so we can count the number of mosaic floorplans by counting the number of the twin binary trees.

THEOREM 2.1.16. *The number of distinct mosaic floorplans with n blocks is equal to the nth Baxter number:*

$$B(n) = \binom{n+1}{1}^{-1} \binom{n+1}{2}^{-1} \sum_{k=1}^{n} \binom{n+1}{k-1} \binom{n+1}{k} \binom{n+1}{k+1}.$$

PROOF. Dulucq and Guibert [1998] proved that the exact number of distinct twin binary trees with n nodes is the Baxter number $B(n)$[Baxter 1964]. Hence, from Theorem 2.1.15, we can deduce that the number of distinct mosaic floorplans with n blocks is equal to $B(n)$. □

2.2 The Exact Number of Slicing Floorplan Configurations

A slicing floorplan was usually represented by a binary tree, which can be obtained by recursively cutting a rectangle into exactly two parts by either a vertical or a horizontal line. This representation is intuitive and simple to implement. However, there is redundancy in this representation. For example, the floorplan in Figure 11 can be mapped to two different binary trees.

In order to find the exact number of different slicing floorplans of n blocks, we adopt another representation called a *v-h tree* [Szepieniec and Otten 1980].

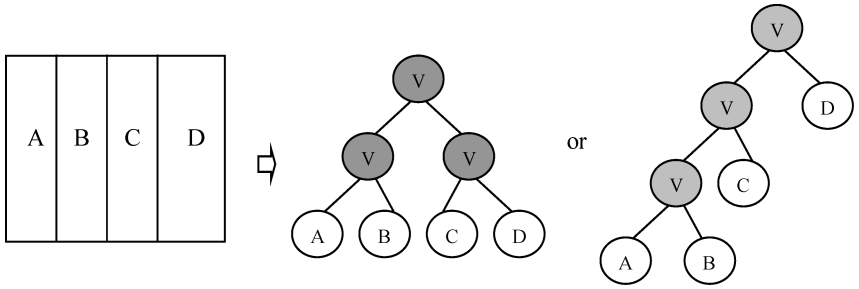


Fig. 11. Example of redundancy in a slicing tree representation.

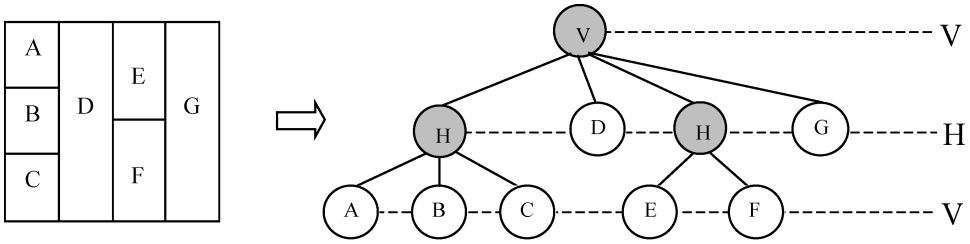


Fig. 12. A v-h tree representation of a slicing floorplan.

We restate the definition of a v-h tree representation for a slicing floorplan below:

Definition 2.2.1 (*A v-h Tree Representation of a Slicing Floorplan*). A tree with the following properties is called a v-h tree of a slicing floorplan with n blocks:

(1) The tree is a rooted ordered tree with each node representing a rectangle in the floorplan.
(2) The rectangles obtained by slicing a rectangle are represented as the children of the node representing the sliced rectangle. These children are ordered according to their relative position.
(3) The root of the tree is the whole floorplan rectangle. The n leaves represent the n blocks, which are not divided further into smaller rectangles in the floorplan.
(4) The nodes in the same level of the tree form a generation. The generations are formed alternatively by vertical and horizontal partition lines. We assign a label V to each generation corresponding to vertically partitioned rectangles, and a label H to each generation corresponding to horizontally partitioned rectangles.
(5) To each leaf node, we assign the label of its corresponding block to it. To each internal node, we assign the label of the generation it belongs to.

Figure 12 gives an example of a slicing floorplan and its corresponding v-h tree representation. The root of the v-h tree is labeled *V with four children*. Correspondingly, the chip is divided into four vertical strips. The left-most child

of the root is then horizontally partitioned into three blocks. A v-h tree is an ordered tree and thus can be represented naturally as a binary tree. A v-h tree differs from the trees in both twin binary trees representations and O-Tree representations in the sense that, in a v-h tree, only the leaf nodes represents the blocks, while in the trees for twin binary trees and O-Trees each node represents a block in the floorplan.

There is one-to-one correspondence between the v-h tree representation and the slicing floorplan [Szepieniec and Otten 1980]. Therefore, we can count the number of the slicing floorplans by counting the number of v-h trees. We have the following theorem regarding the number of the slicing floorplans.

THEOREM 2.2.2. *The number of slicing floorplans with $n > 1$ blocks is twice the Schröder number A_n, defined by*:

$$A_0 = 1; A_1 = 1;$$
$$A_n = (3(2n-3)A_{n-1} - (n-3)A_{n-2})/n.$$

PROOF. There are two different labels, V or H, for the root of a v-h tree. Thus, from Etherington [1940] we see that the number of slicing floorplans with $n > 1$ blocks is twice the Schröder number A_n. □

3. RELATIONSHIPS BETWEEN REPRESENTATIONS

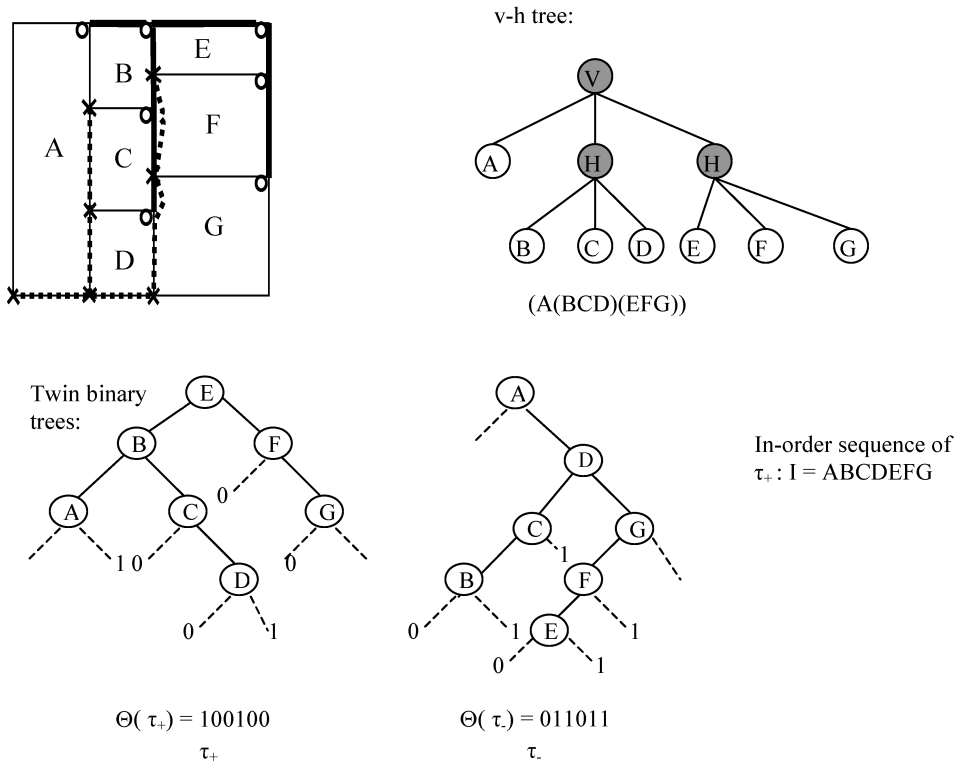
In this section, we describe the relationships between the following four kinds of representations: twin binary trees, sequential pair, corner block list, and O-Tree.

To illustrate the relationships between these representations, we use three examples: a slicing floorplan (Figure 13), a mosaic floorplan (Figure 15), and a general floorplan (Figure 16).

3.1 Slicing Floorplan Example

The slicing floorplan in Figure 13 is first dissected into three strips vertically: the left strip contains block A; the middle strip contains blocks B, C, and D; and the right strip contains blocks E, F, and G. Second, the middle and the right strips are cut horizontally into three pieces. The v-h tree representation is shown in Figure 13. In Figure 13 we also include the twin binary trees representation. The upper-right corner of each block is marked with a circle and the bottom-left corner is marked with a cross. The upper-right corners are connected to their C^+-parents with solid lines. These solid lines and circles form the τ_+ of the twin binary trees. Similarly, the bottom-left corners are connected to their C^--parents by dashed lines. These dashed lines and crosses form the τ_- of the twin binary trees.

Given a floorplan and the dimensions of the blocks, we can find a sequence pair $SP = (S_1, S_2)$ to represent the floorplan [Murata et al. 1995]. The S_1 of the SP is the order of the blocks from upper-left to the right-bottom, and S_2 is the order of the blocks from bottom-left to upper-right. Thus, many SPs may correspond to the same floorplan. For example, in Figure 13, block F can either be above block D or to the right of block D. Thus, we have different SP representations



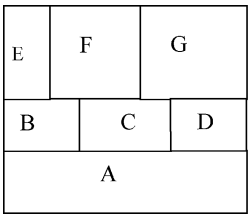
One SP representation of this floorplan is $SP_1=(S_1, S_2)$= (ABECFDG, ADCGBFE). Also the sequence pair $SP_2=(S_1, S_2)$= (ABCDEFG, ADCGBFE) and $SP_3=(S_1, S_2)$= (ABCDEFG, ADCBGFE) represent the same floorplan.

CB: S = (ADCBGFE), L = (100100), T=(00011000)

Fig. 13. Different representations for a slicing floorplan.

(SP_1, SP_2, SP_3) (Figure 13). With the given block dimensions, the three SPs, SP_1, SP_2, and SP_3, produce the same floorplan.

For the floorplan in Figure 13, there are quite a few different O-tree representations. For example, block F is right adjacent to blocks B and C; thus, node F can be the child of either B or C in an O-tree representation. Two different



Corner block List
representation of F^{90°:
S = (ABCDEFG);
L = (111011);
T = (00011000).

Fig. 14. Rotation of the floorplan in Figure 15.

O-tree representations are shown in Figure 14. An O-tree can be mapped to a binary tree by converting the sibling relations to left child branches and converting the descendent relations to right child branches of a binary tree [Chang et al. 2000]. In Figure 13, O-tree OT2 and the τ_- of the twin binary trees are identical after the tree conversion.

For further discussion, we define the 90° rotation of a floorplan as follows:

Definition 3.1 (90° *Rotation of a Floorplan*). We use F^{90} to denote the floorplan obtained by rotating floorplan F, 90° counterclockwise.

For example, Figure 14 shows the F^{90° rotation of the floorplan F in Figure 13. The CB (S, L, T) representation of F^{90} has S = ABCDEFG (Figure 14), which is identical to $I(\tau)$, the order of the twin binary trees representation (τ_+, τ_-) of F (Figure 13).

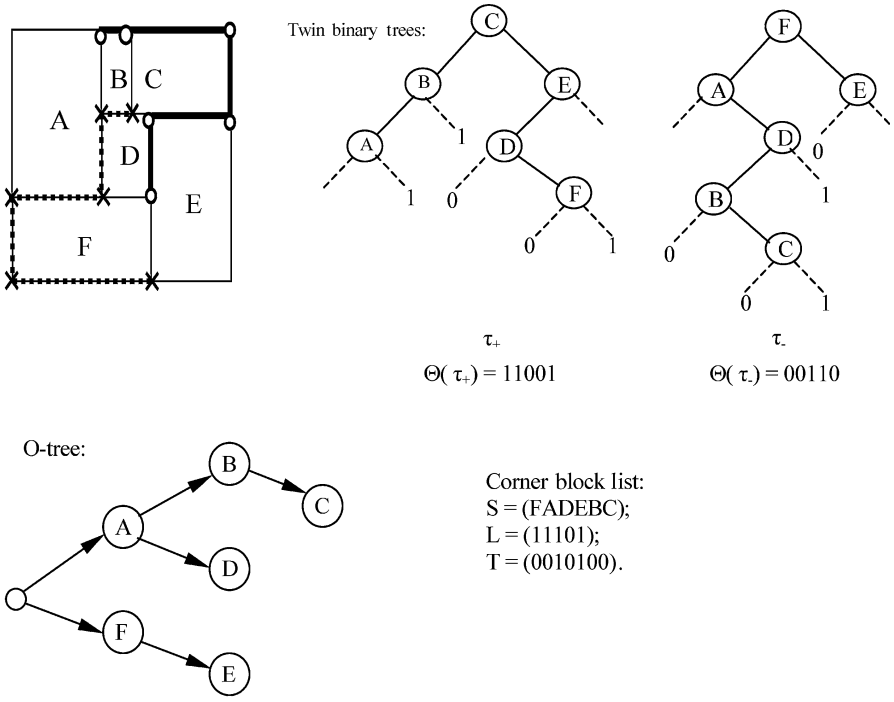
We traverse τ_+ of the twin binary trees in Figure 13 in a depth-first order and get the in-order sequence of τ_+: $I(\tau_+)$ = ABCDEFG. This sequence is the same as the first sequence of SP_2. The corner block list representation, CB = (S, L, T), is given at the bottom of Figure 13. We note that the block sequence S = ADCBGFE of CB is identical to the second sequence of SP_3.

3.2 A Mosaic Floorplan Example

Figure 15 describes an example of a mosaic floorplan. A v-h tree is not available to represent this kind of floorplan. We illustrate the other four representations. The twin binary trees are marked by circles and crosses as shown in Figure 15. Two SPs, SP1 and SP2, out of many possible choices, are described in the figure. The in-order traversal of the τ_+ in the twin binary trees representation produces the sequence $I(\tau_+)$ = ABCDFE, which is same as the first sequence of SP_1 and SP_2. In Figure 15, an O-tree representation is also given. Its binary tree representation is identical to the τ_- of the twin binary trees after tree conversion. The CB representation is next to the O-tree representation in Figure 15. Its sequence S = FADEBC is same as the second sequence of SP_1.

3.3 A General Floorplan Example

Figure 16 illustrates a general floorplan. Only the O-tree and the sequence pair are capable of representing this general floorplan. The O-tree and SP representations are shown in the figure.



Sequence Pair: $SP_1 = (S_1, S_2) = (ABCDFE, FADEBC)$. Also $SP_2 = (S_1, S_2) = (ABCDFE, FADBEC)$ refers to the same floorplan.

Fig. 15. A mosaic floorplan and its different representations.

3.4 Statements of the Relationships

According to the observations in the above examples, we can derive the following relationships as shown in Figure 17:

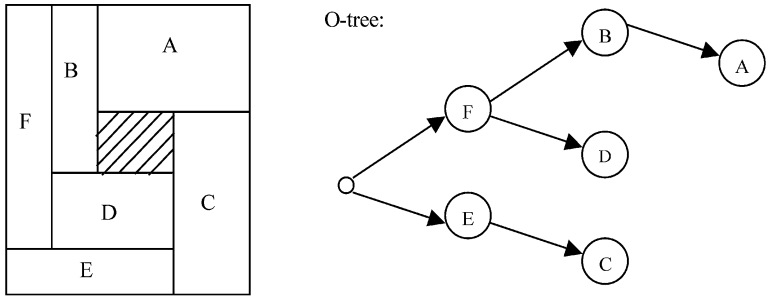
LEMMA 3.1.1. *Given a mosaic floorplan F, the order of its corresponding twin binary trees $TBT(\tau_+, \tau_-)$, $I(\tau_+)$, is identical to the sequence S in the corner block list $CB = (S, L, T)$ of the floorplan F*[90].

PROOF. See Appendix 2. □

LEMMA 3.1.2. *Given a mosaic floorplan and its corresponding corner block list $CB = (S, L, T)$, there exists a sequence pair $SP = (S_1, S_2)$ corresponding to the same floorplan such that the second sequence of SP is same as the sequence S of the corner block list.*

PROOF. See Appendix 3. □

LEMMA 3.1.3. *Given a mosaic floorplan and its corresponding twin binary trees (τ_+, τ_-), there exists a sequence pair $SP = (S_1, S_2)$ corresponding to the same floorplan such that the first sequence of SP is the same as $I(\tau_+)$.*



Sequence pair: $SP_1 = (\Gamma_+, \Gamma_-) = (FBADEC, EFDBCA)$.
Also, $SP2 = (\Gamma_+, \Gamma_-) = (FBADEC, EDFBCA)$.

Fig. 16. A general floorplan with its O-tree and SP representations.

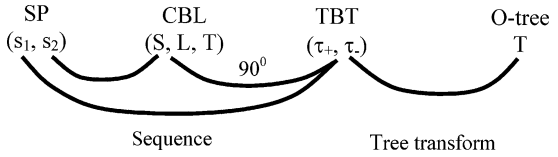


Fig. 17. Summary of the relationships between floorplan representations.

PROOF. This follows by applying Lemmas 3.1.1 and 3.1.2. □

LEMMA 3.1.4. *Given a mosaic floorplan and its corresponding twin binary trees TBT* (τ_+, τ_-), *there exists an O-tree corresponding to the same floorplan such that* τ_- *is identical to the O-tree after the tree conversion from a binary tree to an ordered tree.*

PROOF. Suppose that node i is the right child of node j in binary tree τ_-. Then from line 10 of algorithm MFTB, we see that block i is adjacent to the right-hand edge of block j. Thus, we have node i as a child of node j in the O-tree representation.

Suppose node i is the left child of node j in the binary tree τ_-. Then from line 11 of algorithm MFTB, we derive that block i is adjacent to the top edge of block j. Thus, we have node i as the sibling of node j in the O-tree representation.

The above two cases convert the binary tree τ_- to an ordered tree, and they are clearly representations of the same floorplan. This completes the proof. □

4. CONCLUSIONS

We have shown that the numbers of mosaic floorplan and slicing floorplan configurations are Baxter numbers and Schröder numbers, respectively. A new tree structure representation, called *twin binary trees*, is proposed to represent mosaic floorplans. This representation is concise in the sense that the transformation between the representation and the floorplan is one-to-one. We also identify the relationships between several representations, where the proposed twin binary trees serve as a bridge for these relationships.

APPENDIX 1: PROOF OF THEOREM 2.1.11

THEOREM 2.1.11. *The MFTB algorithm produces a pair of binary trees* (τ_+, τ_-) *satisfying two properties*:

- *Order of the Tree:* $I(\tau_+) = I(\tau_-)$.
- Θ*-labeling:* $\Theta(\tau_+) = \Theta^C(\tau_-)$.

PROOF. We prove the theorem by induction on the number of blocks. When the number of blocks is 1, the statement is true because both τ_+ and τ_- contain only one node and both $\Theta(\tau_+)$ and $\Theta(\tau_-)$ are empty strings. Assume the statement is true when the number of blocks is n. We show that the statement is also true when the number of blocks is $n+1$.

We follow the block deletion operation proposed by Hong et al. [2000]. An $(n+1)$-block floorplan can be reduced to an n-block floorplan by deleting the block at its upper-right corner. Note that for a mosaic floorplan, the block at the upper-right corner of the floorplan is unique. We call this block the *corner block* of the floorplan. There are two orientations of the T-junction at the lower-left end of the corner block, that is, 90° and 180°. Corresponding to these two orientations of the T-junction, Hong et al. [2000] proposed H-deletions and V-deletions. To simplify the discussion, we assume the operation is an H-deletion (a T-junction with a 180° orientation) (Figure 18(a)). The argument for a V-deletion is similar.

Let us denote the corner block as block A; block B is left of block A at the lower-left corner of block A, the floorplan of $n+1$ blocks by F^P, and the floorplan of n blocks (after H-deletion) by F. The corresponding binary tree pairs produced by the MFTB algorithm are $({\tau_+}^P, {\tau_-}^P)$ for floorplan F^P and (τ_+, τ_-) for floorplan F. By the induction assumption, the pair of binary trees (τ_+, τ_-) satisfies the "order of the tree" and the twin binary trees properties.

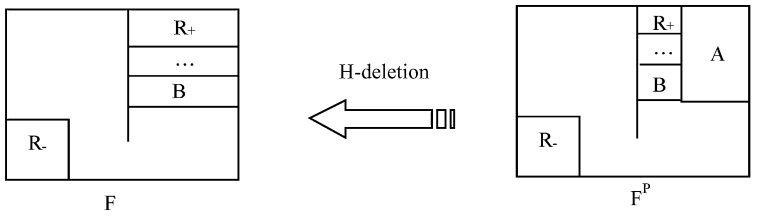
Figure 18 shows the relationships of the binary trees τ_+ and ${\tau_+}^P$ and of the binary trees τ_- and ${\tau_-}^P$. We list the properties of binary trees ${\tau_+}^P$, τ_+, ${\tau_-}^P$ and τ_-, as follows:

Properties of the binary tree ${\tau_+}^P$: include the following:

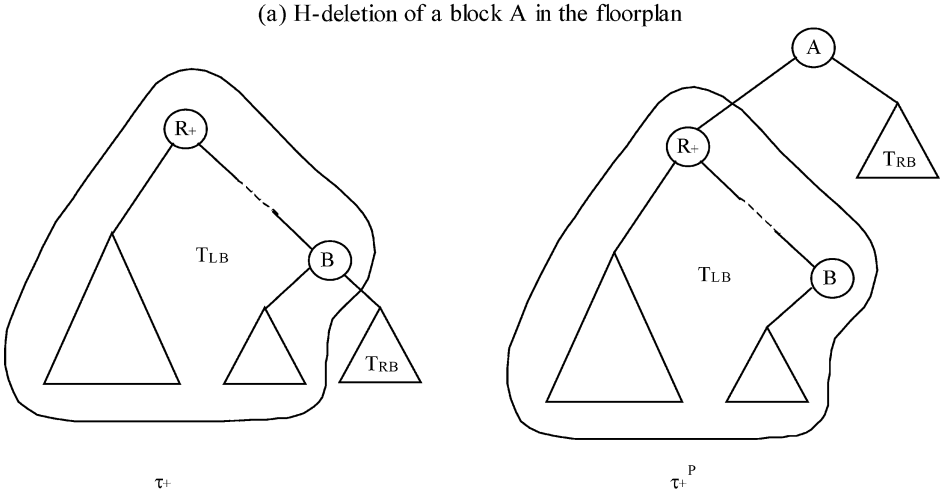
(1) Node A is the root of binary tree ${\tau_+}^P$. The root branches out to two subtrees, left subtree T_{LB} and right subtree T_{RB}. We use node R_+ denote the root of subtree T_{LB}.
(2) Node B is in subtree T_{LB}.
(3) Node B has no right child because the lower-right end of block B has a T-junction in a 180° orientation.

Properties of the binary tree τ_+ include the following:

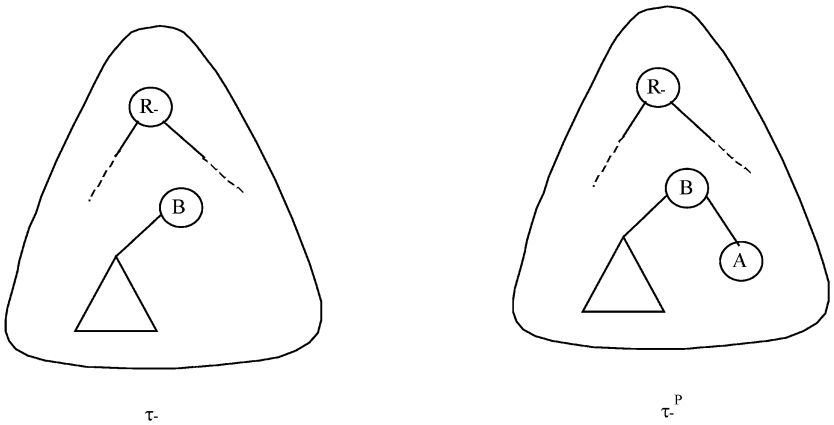
(1) Binary tree τ_+ is similar to binary tree ${\tau_+}^P$ except that node A is deleted. Node B inherits the right subtree T_{RB} of node A.
(2) Node B is a descendent of root R_+ through consecutive right branches because block B is at the right edge of the floorplan (Figure 18(b)).



(a) H-deletion of a block A in the floorplan



(b) Transformation from τ_+ to τ_+^P under H-deletion



(c) Transformation from τ_- to τ_-^P under H-deletion

Fig. 18. The H-deletion and the corresponding transformation of τ_+ and τ_-.

The property of the binary tree $\tau_-{}^P$ is as follows:

Node A is a leaf node and a right child of node B (Figure 18(c)).

The property of the binary tree τ_- is as follows:

Binary tree τ_- is identical to binary tree τ_-^{P} except that node A is deleted (Figure 18(c)). Thus, node B has no right child.

The proof of order of the tree property goes as follows. We show the relations (i) between $\mathrm{I}(\tau_+{}^{\mathrm{P}})$ and $\mathrm{I}(\tau_+)$, and (ii) between $\mathrm{I}(\tau_-{}^{\mathrm{P}})$ and $\mathrm{I}(\tau_-)$, hold. By our induction assumption, $\mathrm{I}(\tau_+)$ and $\mathrm{I}(\tau_-)$ are identical. From (i) and (ii), we can conclude that $\mathrm{I}(\tau_+{}^{\mathrm{P}})$ and $\mathrm{I}(\tau_-{}^{\mathrm{P}})$ are identical.

(i) $\mathrm{I}(\tau_+{}^{\mathrm{P}})$ is identical to $\mathrm{I}(\tau_+)$ except that A is inserted to the right of B. Node A is the next of node B in the sequence of the traversal because in binary tree $\tau_+{}^{\mathrm{P}}$, node B has no right child (Property 3 of $\tau_+{}^{\mathrm{P}}$), node B is a descendent of node R_+ via consecutive right branches (Properties 1 and 2 of τ_+), and node A is the parent of node R_+ (Property 1 of $\tau_+{}^{\mathrm{P}}$). (Figure 18b)).

(ii) $\mathrm{I}(\tau_-{}^{\mathrm{P}})$ is identical to $\mathrm{I}(\tau_-)$ except that A is inserted to the right of B: the statement is true because node A is the right child of node B in binary tree $\tau_-{}^{\mathrm{P}}$. (Figure 18(c)).

The proof of the Θ-labeling property goes as follows. We first decompose $\Theta(\tau_+)$ and $\Theta(\tau_-)$ according to subtrees. We then write $\Theta(\tau_+{}^{\mathrm{P}})$ and $\Theta(\tau_-{}^{\mathrm{P}})$ using the same notation. Finally, we show $\Theta(\tau_+{}^{\mathrm{P}}) = \Theta^{\mathrm{C}}(\tau_-{}^{\mathrm{P}})$ from the assumption that $\Theta(\tau_+) = \Theta^{\mathrm{C}}(\tau_-{}^{\mathrm{P}})$.

(1) We decompose $\Theta(\tau_+)$ into the following expression (Figure 18(b)):

$$\Theta(\tau_+) = \Theta(\tau_+, \mathrm{B})0\Theta(\mathrm{T_{RB}}). \tag{2.1}$$

The bit 0 is inserted between $\Theta(\tau_+, \mathrm{B})$ and $\Theta(\mathrm{T_{RB}})$ because the bit 0 is the first label of tree $\mathrm{T_{RB}}$ (property 1 of Θ-labeling).

(2) We decompose $\Theta(\tau_-)$ into the following expression (Figure 18(c)):

$$\Theta(\tau_-) = \Theta(\tau_-, \mathrm{B})1\Theta_r(\tau_-). \tag{2.2}$$

The bit 1 is appended right after $\Theta(\tau_-, \mathrm{B})$ because node B has no right child (Property 1 of tree τ_-). We use $\Theta_r(\tau_-)$ to denote the rest of the sequence in $\Theta(\tau_-)$.

(3) We describe $\Theta(\tau_+{}^{\mathrm{P}})$ as follows:

$$\Theta(\tau_+{}^{\mathrm{P}}) = \Theta(\tau_+, \mathrm{B})10\Theta(\mathrm{T_{RB}}). \tag{2.3}$$

The bit 1 is inserted between $\Theta(\tau_+, \mathrm{B})$ and $0\Theta(\mathrm{T_{RB}})$ (Equation (2.1)) because node B has no right child in $\tau_+{}^{\mathrm{P}}$ (Property 3 of tree $\tau_+{}^{\mathrm{P}}$).

(4) We express $\Theta(\tau_+{}^{\mathrm{P}})$ as follows:

$$\Theta(\tau_-{}^{\mathrm{P}}) = \Theta(\tau_-, \mathit{B})01\Theta_r(\tau_-). \tag{2.4}$$

The bit 0 is inserted between $\Theta(\tau_-, \mathrm{B})$ and $1\Theta_r(\tau_-)$ (Equation (2.2)) because node S has no left child in $\tau_-{}^{\mathrm{P}}$ (Property 1 of tree $\tau_-{}^{\mathrm{P}}$).

(5) By our induction assumption, the two strings $\Theta(\tau_+)$ and $\Theta(\tau_-)$ are bit-wise complements. From Formulas (2.1) and (2.2), we have $\Theta(\tau_+, \mathrm{B})0\Theta(\mathrm{T_{RB}}) =$

$\Theta^{C}(\tau_-, B)0\Theta^{C}(\tau_-)$. Because $|\Theta(\tau_+, B)| = |\Theta^{C}(\tau_-, B)|$ (Property 3 of Θ-labeling), we find that $\Theta(\tau_+, B)10\Theta(T_{RB}) = \Theta^{C}(\tau_-, B)10\Theta^{C}(\tau_-)$.

Thus, from Formulas (2.4) and (2.3), we see that $\Theta({\tau_+}^{P}) = \Theta^{C}({\tau_-}^{P})$.

The Special Case That Subtree T_{RB} Is an Empty Set. For such a case, in trees τ_+ and τ_-, node B is the last node traversed. We have $\Theta(\tau_+) = \Theta(\tau_+, B)$ and $\Theta(\tau_-) = \Theta(\tau_-, B)$. Therefore, from Properties 1 and 2 of tree τ_+, we have $\Theta({\tau_+}^{P}) = \Theta(\tau_-, B)1$, and from Figure 9(c), we have $\Theta({\tau_-}^{P}) = \Theta(\tau_-, B)0$. These two equalities conclude the proof. □

APPENDIX 2: PROOF OF LEMMA 3.1.1

LEMMA 3.1.1. *Given a mosaic floorplan F, the order $I(\tau_+)$ of its corresponding twin binary trees TBT (τ_+, τ_-) is identical to the sequence S in the corner block list $CB(S, L, T)$ of floorplan F^{90}.*

PROOF. The proof will be given by induction on the number of blocks.

When the number of blocks is 1, the statement is true, since both floorplan F and floorplan F^{90} have only one block and $I(\tau_+)$ and S both contain the only block in the sequence, so $I(\tau_+) = S$.

Assume the statement is true when the number of blocks is n. We show that the statement is also true when the number of blocks is $n + 1$.

We follow the block deletion operation proposed by Hong et al. [2000]. Given an $(n+1)$-block floorplan $(F^{P})^{90}$, we get an n-block floorplan F^{90} by block deletion. There are two kinds of deletions: H-deletions and V-deletions.

The case of V-deletion is shown in Figure 19(a). After deleting the right corner block A, floorplan $(F^{P})^{90}$ is transformed to floorplan F^{90}. Thus, the S sequence in the corner block representation of floorplan $(F^{P})^{90}$ is the concatenation of the S sequence for floorplan F and block A.

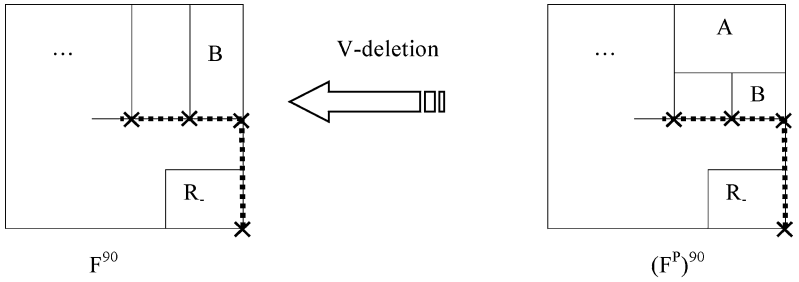
In the figure, we marked the edges and nodes of the binary tree τ_- of the twin binary trees representation for F and F^{P}. The edges of tree τ_- are marked by bold dashed lines and the nodes are marked by crosses. Using the algorithm MFTB, we know that block R_- is the root of the binary tree τ_-^{P}, and it remains to be the root of binary tree τ_-. Block A is the right-most descendant of the trees ${\tau_+}^{P}$ and ${\tau_-}^{P}$, where $({\tau_+}^{P}, {\tau_-}^{P})$ is the twin binary representation of F^{P}. After the deletion of block A, ${\tau_+}^{P}$ remains the same except that node A is removed. Thus, $I({\tau_+}^{P})$ is the concatenation of $I(\tau_+)$ and block A.

By the induction assumption, $I(\tau_+)$ is identical to the S sequence for F. Therefore, $I(\tau^{P})$ is identical to the S sequence for F^{P}. Hence, the statement is true for $n + 1$ blocks.

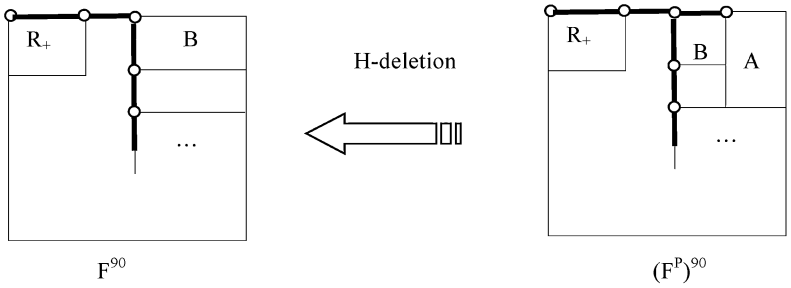
The case for H-deletion is similar to the V-deletion case except that we consider the change of τ_-^{P} in the process of deletion (Figure 19(b)). The statement is also true for this case. This completes the proof. □

APPENDIX 3: PROOF OF LEMMA 3.1.2

LEMMA 3.1.2. *Given a mosaic floorplan and its corresponding corner block list $CB = (S, L, T)$, there exists a sequence pair $SP = (S_1, S_2)$ corresponding to*



(a) A V-deletion of a block S to the floorplan



(b) An H-deletion of a block S to the floorplan

Fig. 19. H-and V-deletions and the corresponding transformations of τ_+ and τ_-.

the same floorplan such that the second sequence of SP is same as the sequence S of the corner block list.

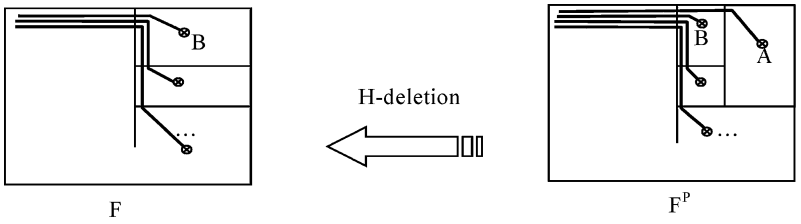
PROOF. When the number of blocks is 1, the statement is true because both the sequence S and the sequence S_2 contain the only block in the floorplan.

Assume the statement is true for the floorplan F with n blocks. We show that the statement is also true for the floorplan F^P with $n + 1$ blocks.

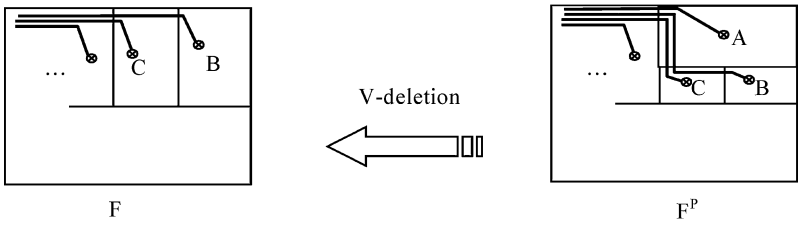
We use the deletion technique similar to the one used in the proof of Lemma 3.1.1.

The case of H-deletion is shown in Figure 20(a). We marked the negative locus for each block in the figure, which was used in Murata et al. [1995] to derive the second sequence of the sequential pair. The H-deletion will not change other blocks' loci, except that the most outside locus for block A is removed. This implies the S_2 for floorplan F^P is identical to the S_2 for floorplan F with block A appended. By the definition of the corner block list, the S sequence for floorplan F^P is identical to the S sequence for floorplan F with block A appended. Thus, by the induction assumption, the S sequence is identical to the S_2 sequence for floorplan F^P.

In the case of V-deletion (Figure 20(b)), only the loci of the blocks right below block A (blocks B and C in Figure 20(b)) are modified. However, the order of the loci is kept since they never cross each other. Therefore, the statement is also true by similar reasoning as in the H-deletion case. This completes the proof. □



(a) H-deletion of a block S to the floorplan



(b) V-deletion of a block S to the floorplan

Fig. 20. The H- and V-deletions and the corresponding transformations of loci.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers for their comments.

REFERENCES

BAXTER, G. 1964. On fixed points of the composite of commuting functions. *Proc. Amer. Math. Soc. 15*, 6, 851–855.

CHANG, Y., CHANG, Y., WU, G., AND WU, S. 2000. B*-trees: A new representation for non-slicing floorplans. In *Proceedings of the 37th Design Automation Conference*. 458–463.

CHUNG, F. R. K., GRAHAM, R. L., HOGGATT, JR., V. E., AND KLEIMAN, M. 1978. The number of Baxter permutations. *J. Combin. Theor.*, Series A, *24*, 3 (May), 382–394.

DULUCQ, S. AND GUIBERT, O. 1998. Baxter permutations. *Discrete Math. 180*, 1–3, 143–156.

ETHERINGTON, I. M. H. 1940. Some problems of non-associative combinations. *Edinburgh Math. Notes 32*, pp. i–vi.

GRASON, J. 1970. *A Dual Linear Graph Representation for Space-filling Location Problems of the Floor-Planning Type*. MIT Press, Cambridge, MA.

GUO, P., CHENG, C. K., AND YOSHIMURA, T. 1999. An O-tree representation of non-slicing floorplan and its applications. In *Proceedings of the 36th Design Automation Conference*. 268–273.

HONG, X., ET AL. 2000. Conner block list: An effective and efficient topological representation of non-slicing floorplan. In *Proceedings of the International Conference on Computer Aided Design (ICCAD '00)*. 8–12.

KNUTH, D. E. 1997. *The Art of Computer Programming*. Addison-Wesley, Reading, MA.

KOZMINSKY, K. AND KINNEN, E. 1985. Rectangular duals of planar graph. *Networks*, *15*, 2, 145–157.

MURATA, H., FUJIYOSHI, K., NAKATAKE, S., AND KAJITANI, Y. 1995. Rectangle-packing-based module placement. In *Proceedings of the International Conference on Computer Aided Design*. 472–479.

NAKATAKE, S., FUJIYOSHI, K., MURATA, H., AND KAJITANI, Y. 1998. Module packing based on the BSG-structure and IC layout applications. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. 17*, 6 (June). 519–530.

OTTEN, R. H. J. M. 1982. Automatic floorplan design. In *Proceedings of the ACM/IEEE Design Automation Conference*. 261–267.

OTTEN, R. H. J. M. 2000. What is a Floorplan? In *Proceedings of the International Symposium on Physical Design*. 212–217.

SAKANUSHI, K. AND KAJITANI, Y. 2000. The quarter-state sequence (Q-sequence) to represent the floorplan and applications to layout optimization. In *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems*. 829–832.

SZEPIENIEC, A. A. AND OTTEN, R. H. J. M. 1980. The genealogical approach to the layout problem. In *Proceedings of the 17th Design Automation Conference*. 535–542.

TANG, X. AND WONG, M. 2001. FAST-SP: A fast algorithm for block placement based on sequence Pair. In *Proceedings of Asia and South Pacific Design Automation Conference*. (Yokohama, Japan. Jan.). 521–526.

Received August 2001; accepted May 2002