Corner Sequence—A P-Admissible Floorplan Representation With a Worst Case Linear-Time Packing Scheme

Jai-Ming Lin, Yao-Wen Chang, *Member, IEEE*, and Shih-Ping Lin

***Abstract*—Floorplanning/placement allocates a set of modules into a chip so that no two modules overlap and some specified objective is optimized. To facilitate floorplanning/placement, we need to develop an efficient and effective representation to model the geometric relationship among modules. In this paper, we present a P-admissible representation, called corner sequence (CS), for nonslicing floorplans. CS consists of two tuples that denote the packing sequence of modules and the corners to which the modules are placed. CS is very effective and simple for implementation. Also, it supports incremental update during packing. In particular, it induces a generic worst case linear-time packing scheme that can also be applied to other representations. Experimental results show that CS achieves very promising results for a set of commonly used MCNC benchmark circuits.**

***Index Terms*—Floor planning, layout, physical design, placement, VLSI design.**

I. Introduction

As TECHNOLOGY advances, circuit sizes and design complexity increase rapidly. To cope with the increasing design complexity, hierarchical design and intellectual property (IP) modules are widely used. Further, as device dimensions are reduced, the capacitive, resistive, and inductive parasitic effects increase, which makes interconnect delay become the dominant factor in determining the overall circuit performance in deep submicrometer technologies. As an early stage of very large scale integration (VLSI) physical design, floorplanning has a great impact on chip size and global interconnect structure. This trend makes module floorplanning/placement much more critical to the quality of a VLSI design than ever. To facilitate floorplanning/placement, however, we need a representation to model the geometric relationship among modules. Such a representation induces a solution structure for floorplan optimization. It is thus desired to develop an efficient, flexible, and effective representation to cope with the high complexity in modern floorplan design.

Manuscript received February 2, 2002; revised September 6, 2002. This work was supported in part by the National Science Council of Taiwan, R.O.C., under Grant NSC-89-2215-E-009-117.

J.-M. Lin was with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. He is now with the Realtek Semiconductor Corporation, Hsinchu 300, Taiwan, R.O.C.

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

S.-P. Lin is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: gis89583@cis.nctu.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2003.816137

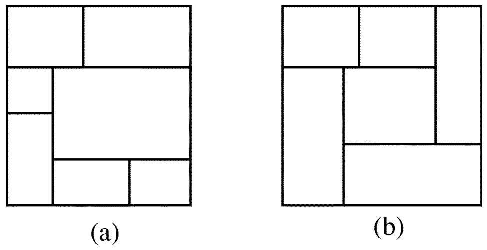

Fig. 1. (a) Slicing floorplan. (b) Nonslicing floorplan.

A. Previous Work

There exist a few floorplan representations in the literature, e.g., [1], [3], [4], [7]–[11], and [15]. We shall first review these representations and the type of floorplans that they can represent. A slicing floorplan is one of the simplest types of floorplans. A slicing structure can be obtained by recursively cutting rectangles horizontally or vertically into smaller rectangles; otherwise it is nonslicing. Fig. 1(a) and (b) shows a slicing floorplan and a nonslicing one, respectively. For the slicing structure, Otten [11] first used a binary tree to represent a slicing floorplan. Wong and Liu [15] proposed a normalized Polish expression to improve the binary-tree representation. The slicing structure has smaller solution space, resulting in faster running time. However, most designs have nonslicing floorplan structures.

For the nonslicing floorplan structure, there exist several well-known "old" graph-based representations. Ohtsuki *et al.* [10] used a pair of horizontal and vertical directed acyclic graphs, called *polar graphs*, to represent a topological placement. Other representations such as *adjacency graphs* and *channel intersection graphs* are also widely used [13].

The nonslicing floorplan representations have attracted much attention in the literature recently, e.g., *sequence pair (SP)* [8], *bounded-sliceline grid (BSG)* [9], *O-tree* [3], *B^*-tree* [1], *corner block list (CBL)* [4], and *transitive closure graph (TCG)* [7]. Murata *et al.* [8] used two sequences of module names (called the SP) to represent the geometric relations among modules. They defined the *P-admissible solution space*, which satisfies the following four requirements [8]:

1) solution space is finite;
2) every solution is feasible;
3) packing and cost evaluation can be performed in polynomial time;

4) best evaluated packing in the space corresponds to an optimal placement.

(By this definition, the slicing tree is not a P-admissible representation since it cannot represent many optimal nonslicing placements.) For cost evaluation, the SP needs to compute longest paths in its induced constraint graphs, which takes significant running time. To reduce the time complexity, Tang and Wong [14] proposed a faster packing scheme (called *FAST-SP*) by computing the longest common subsequence. Another nonslicing representation, namely, the BSG, was proposed by Nakatake *et al.* [9]. There could be multiple representations corresponding to one BSG packing and, thus, the BSG incurs significant redundancies. Both the SP and BSG are P-admissible and can represent general floorplans.

Guo *et al.* [3] first proposed a tree-based representation (called the O-tree) for *compacted* nonslicing floorplans. To obtain a good solution after perturbation, a sequence of one-dimensional compaction and transformations between a placement and its representation are required. Chang *et al.* [1] presented a binary tree-based representation (called the B*-tree), also for *compacted* nonslicing floorplans. The B*-tree is a restricted version of the O-tree with faster operations and simpler data structures. Given a B*-tree, however, it may fail to obtain a placement corresponding to the original B*-tree because of its two-dimensional packing nature.

Hong *et al.* [4] proposed a CBL for *mosaic* nonslicing floorplans. By mosaic floorplans, we mean that each room (region) in the chip contains one and only one module (i.e., no empty region). The CBL has a faster packing scheme; however, it is not P-admissible since it cannot guarantee a feasible solution in each perturbation and many infeasible solutions may be generated before a feasible solution is found. Further, the mosaic floorplans that the CBL representation can represent are more restricted than compacted floorplans (but are more general than slicing ones).

Recently, Lin and Chang proposed a TCG for *general* floorplans [7]. The TCG is P-admissible. Different from the graph-based representations presented in the early days (e.g., the polar graph, adjacency graph, etc.), the TCG can guarantee its feasibility during perturbation by manipulating the transitive and reduction edges in a TCG, resulting in a good structure for solution perturbation. However, the operations to keep a feasible TCG are relatively complicated.

B. Our Contribution

We present, in this paper, a P-admissible representation [called a *corner sequence* (CS)] for *compacted* nonslicing floorplans. A CS consists of two tuples that denote the packing sequence of modules and the corresponding corners to which the modules are placed. A CS is very effective and very simple for implementation. Also, it supports incremental update during packing. In particular, a CS induces a generic worst case linear-time packing scheme that can also be applied to other existing representations. Experimental results show that a CS obtains the best silicon area and wirelength for a set of commonly used MCNC benchmark circuits, as compared to the published works.

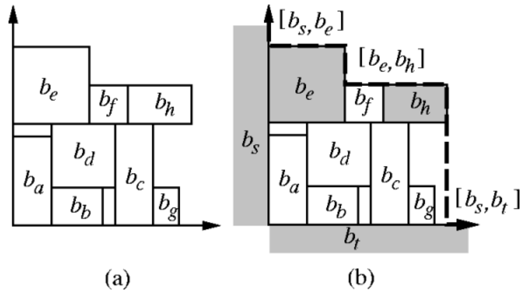

Fig. 2. (a) Placement $\mathcal{P}$ in a chip. (b) The contour R of $\mathcal{P}$.

The remainder of this paper is organized as follows. Section II formulates the floorplan/placement design problem. Section III presents the procedures to build a CS from a placement and transform a CS to a placement. Section IV introduces the perturbations and floorplan design algorithm for a CS. Experimental results are reported in Section V. Finally, we conclude our study and discuss future research directions in Section VI.

II. Preliminaries

Let $B = \{b_1, b_2, \ldots, b_m\}$ be a set of m modules whose width, height, and area are denoted by W_i, H_i, and A_i, $1 \le i \le m$. Let (x_i, y_i) $((x'_i, y'_i))$ denote the coordinate of the bottom–left (top–right) corner of module b_i. A placement $\mathcal{P}$ is an assignment of (x_i, y_i) for each b_i, $1 \le i \le m$, such that no two modules overlap. The goal of floorplanning/placement is to minimize some predefined cost metric such as the area (i.e., the minimum bounding rectangle of $\mathcal{P}$), the wirelength (i.e., the summation of the half-bounding box of interconnections), or a linear combination of area and wirelength, induced by the assignment of b_i's on the chip.

In the CS representation, modules are placed one at a time according to a predefined sequence. When a module is placed, we insert the module into two neighboring modules in the contour formed by those placed modules. To facilitate this process, we should first introduce the concept of the contour. We use the following notations for the technical discussions.

- b_s (b_t): b_s (b_t) denotes the *dummy module* on the left side (bottom) of a placement; it is used to specify the starting (ending) position of a contour in a placement. Let $(x'_s, y'_s) = (0, \infty)$ and $(x'_t, y'_t) = (\infty, 0)$.
- $b_i \preceq_x b_j$: b_i is dominated by b_j in the x direction if and only if $x'_i \le x_j$ and $y'_i \le y'_j$, i.e., b_j is placed to the right of b_i and b_j's top boundary is higher than b_i's.
- $b_i \preceq_y b_j$: We say that b_i is dominated by b_j in the y direction if and only if $y'_i \le y_j$ and $x'_i \le x'_j$, i.e., b_j is placed higher than b_i and b_j's right boundary is right to b_i's.
- R: The contour $R = \langle b_1 b_2, \ldots, \rangle$ gives a *minimal dominating sequence* of modules, in which the modules are not dominated by any other in the x and y directions in a placement, i.e., $\forall b_i \in R$, $\nexists b_j \in \mathcal{P}$, $b_i \preceq_x b_j$, or $b_i \preceq_y b_j$.
- $[b_i, b_j]$: the *bend* formed by the right boundary of b_i and the top one of b_j of two adjacent modules b_i and b_j in R. Given a contour R with n modules, there exist $n-1$ bends.

Fig. 2(a) shows a placement $\mathcal{P}$ with eight modules b_a, b_b, b_c, b_d, b_e, b_f, b_g, and b_h. As shown in Fig. 2(b), b_s (b_t) denotes the dummy module left to (below) $\mathcal{P}$. Since $x'_a \le x_d$ and $y'_a \le y'_d$,

$b_a \preceq_x b_d$. Similarly, $b_b \preceq_y b_d$ since $y'_b \leq y_d$ and $x'_b \leq x'_d$. The contour R of $\mathcal{P}$ consists of the four modules b_s, b_e, b_h, and b_t that are not dominated by any other module. There exist three bends $[b_s, b_e]$, $[b_e, b_h]$, and $[b_h, b_t]$ in R.

III. Corner Sequence (CS)

In this section, we present the CS representation. CS $= \langle (S_1, D_1)(S_2, D_2), \ldots, (S_m, D_m) \rangle$ uses a packing sequence S of the m modules, as well as the corresponding bends D formed by the modules to describe a compacted placement. We refer to each two-tuple (S_i, D_i) $1 \leq i \leq m$ as a *term* of the CS. We first show how to derive a CS representation from a compacted placement.

A. From a Placement to a CS

A module b_i is said to *cover* another b_j if b_i is higher than b_j and their projections in the x axis overlaps or b_i is to the right of b_j and their projections in the y axis overlaps (i.e., $y'_j \leq y_i$, $x'_j > x_i$, and $x_j < x'_i$ or if $x'_j \leq x_i$, $y'_j > y_i$, and $y_j < y'_i$). Given a left and bottom compacted placement (i.e., an admissible placement named in [3]), we first pick the dummy modules b_s and b_t and make $R = \langle b_s b_t \rangle$ for the two chosen modules. The module b_i on the bottom-left corner of $\mathcal{P}$ is picked (i.e., $S_1 = b_i$ and $D_1 = [b_s, b_t]$) since it is the unique module at the bend of R, and the new R becomes $\langle b_s b_i b_t \rangle$. When there exists more than one module at the bends, we pick the left-most module that does not cover other unvisited modules at the bends. Therefore, the module b_j at the bend $[b_s, b_i]$ is picked if b_j exists and b_j does not cover the other unvisited module b_k at the bend $[b_i, b_t]$; otherwise, b_k is picked. This process continues until no module is available. Based on the above procedure, there exits at least one module at a bend of the current R before all modules are chosen since the placement is compacted.

Theorem 1: There exists a unique CS corresponding to a compacted placement.

Proof: To prove that there exists a unique CS corresponding to a compacted placement, we only need to show that the chosen module of our procedure in each iteration is unique. By our definition, there exits at least one module at a bend of current R before all modules are chosen. Without loss of generality, if R is composed of b_s, b_i, b_{i+1}, ..., and b_t, the left-most module, located at a bend of R, which does not cover others, is unique. (If there are several modules at bends and the first module covers others, we can choose the second one. Similarly, we can choose the third if the second covers others, and so on.) Therefore, by repeatedly choosing such a module in a placement until no module is available, we can build a unique CS corresponding to the placement. ■

Fig. 3(a)–(h) shows the process to build a CS from the placement $\mathcal{P}$ of Fig. 2(a). R initially consists of b_s and b_t. Module b_a at the bottom-left corner is chosen first since it is the unique module at the bend of R ($S_1 = b_a$ and $D_1 = [b_s, b_t]$). Fig. 3(a) shows the resulting R (denoted by heavily shaded areas). Similarly, b_b is chosen ($S_2 = b_b$ and $D_2 = [b_a, b_t]$) and the new R is shown in Fig. 3(b). After b_d in Fig. 3(b) is chosen, b_a and b_b are removed from R since $b_a \preceq_x b_d$ and $b_b \preceq_y b_d$ [see Fig. 3(c) for the new R]. As shown in Fig. 3(d), there exist two modules b_f and b_c at bends. Although b_f is left to b_c, we pick b_c first since b_f covers b_c. This process repeats until no module is available, and the resulting CS is shown in Fig. 3(i).

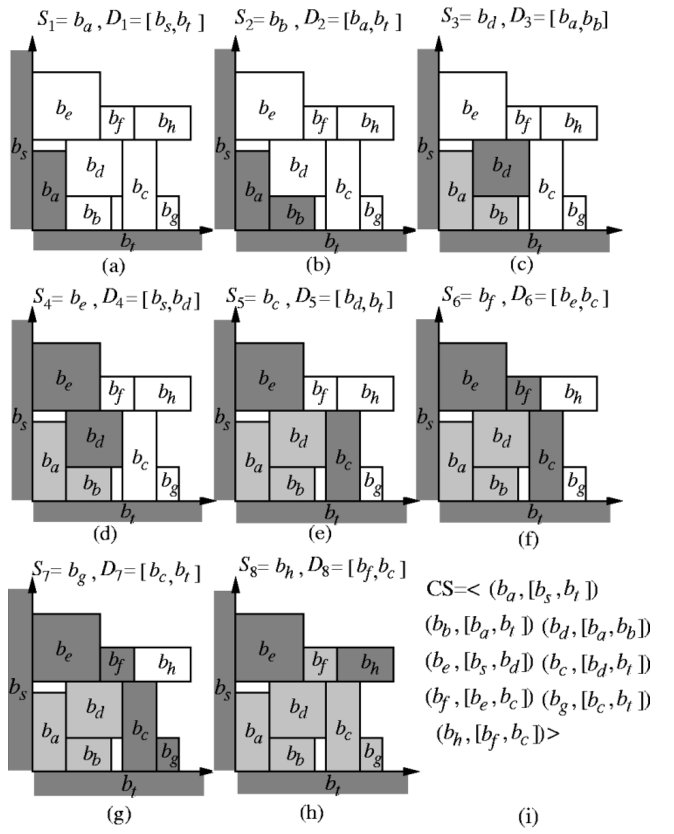
$S_1 = b_a$, $D_1 = [b_s, b_t]$
$S_2 = b_b$, $D_2 = [b_a, b_t]$
$S_3 = b_d$, $D_3 = [b_a, b_b]$
$S_4 = b_e$, $D_4 = [b_s, b_d]$
$S_5 = b_c$, $D_5 = [b_d, b_t]$
$S_6 = b_f$, $D_6 = [b_e, b_c]$
$S_7 = b_g$, $D_7 = [b_c, b_t]$
$S_8 = b_h$, $D_8 = [b_f, b_c]$
b_e b_f b_h b_s b_d b_c b_a b_b b_g b_t
(a) (b) (c) (d) (e) (f) (g) (h) (i)
CS=< $(b_a, [b_s, b_t])$ $(b_b, [b_a, b_t])$ $(b_d, [b_a, b_b])$ $(b_e, [b_s, b_d])$ $(b_c, [b_d, b_t])$ $(b_f, [b_e, b_c])$ $(b_g, [b_c, b_t])$ $(b_h, [b_f, b_c])$>

Fig. 3. (a)–(h) Process to build a CS from a placement. (Note that the heavily shaded modules denote those in R and the lightly shaded ones denote the visited modules.) (i) Resulting CS.

B. From a CS to a Placement

We have introduced the procedure to build a CS from a placement. In this section, we present the packing scheme for a CS [called *dynamic sequence packing* (DSP)], to transform a CS into a placement. For DSP, a contour structure is maintained to place a new module. Let L be a doubly linked list that keeps modules in a contour. Given a CS, we can obtain the corresponding placement in $O(m)$ time by inserting a node into L for each term in the CS, where m is the number of modules.

L initially consists of n_s and n_t that denote dummy modules b_s and b_t, respectively. For each term $(b_i, [b_j, b_k])$, $i = 1, \ldots, m$ in a CS, we insert a node n_i between n_j and n_k in L for b_i and assign the $x(y)$ coordinate of b_i as x'_j (y'_k). This corresponds to placing module b_i at the bend $[b_j, b_k]$. Those modules that are dominated by b_i in the $x(y)$ direction should then be removed from R. This can be done by deleting the predecessor (successor) n_p's of n_i in L if y'_p's (x'_p's) are smaller than $y'_i(x'_i)$. The process repeats until no term in the CS is available. Let $W(H)$ denote the width (height) of a chip. $W = x'_u(H = y'_v)$ if $n_u(n_v)$ is the node right before (behind) $n_t(n_s)$ in the final L.

Fig. 4 shows the packing scheme for the CS shown in Fig. 4(a). L initially consists of n_s and n_t. We first insert a node n_a between n_s and n_t since $S_1 = b_a$ and $D_1 = [b_s, b_t]$. The $x(y)$ coordinate of b_a is $x'_s(y'_t)$. Fig. 4(b) shows the resulting

$$\text{CS}=\langle (b_a,[b_s,b_t])\ (b_b,[b_a,b_t])\ (b_d,[b_a,b_b])\ (b_e,[b_s,b_d])\ (b_c,[b_d,b_t])\ (b_f,[b_e,b_c])\ (b_g,[b_c,b_t])\ (b_h,[b_f,b_c])\rangle$$

(a)

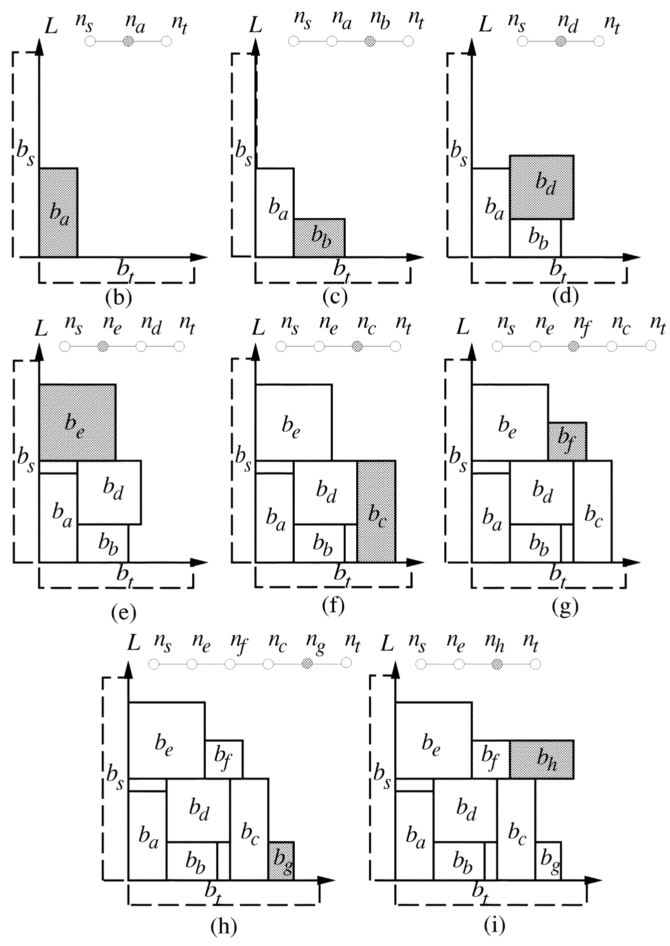

Fig. 4. (b)–(i) DSP packing scheme for the CS shown in (a), where CS $= \langle (b_a,[b_s,b_t])(b_b,[b_a,b_t])(b_d,[b_a,b_b])(b_e,[b_s,b_d])(b_c,[b_d,b_t]),$ $(b_f,[b_e,b_c])(b_g,[b_c,b_t])(b_h,[b_f,b_c])\rangle$.

placement and L. Similarly, n_b is inserted between n_a and n_t in L of Fig. 4(b) since $S_2 = b_b$ and $D_2 = [b_a, b_t]$ [see Fig. 4(c) for the resulting placement and L]. After we insert a node n_d between the two nodes n_a and n_b in L of Fig. 4(c) for the third term (b_d, $[b_a, b_b]$) in the CS, the predecessor n_a (successor n_b) of n_d is deleted because $y'_a \leq y'_d(x'_b \leq x'_d)$ [see Fig. 4(d)]. The process repeats for all terms in the CS, and the resulting placement and L are shown in Fig. 4(i). The width (height) of a chip is $W = x'_h(H = y'_e)$ since the node right before (behind) $n_t(n_s)$ is $n_h(n_e)$ in L.

Theorem 2: The solution space of a CS is bounded by $(m!)^2$, where m is the number of modules.

Proof: CS $= \langle (S_1, D_1)(S_2, D_2), \ldots, (S_m, D_m)\rangle$ uses a packing sequence S formed by m modules, as well as the corresponding bends D of the modules to describe a placement. Given m modules, there are $m!$ permutations in the packing sequence S. By the DSP scheme, we insert a node n_i into L for each module b_i in the CS to get its coordinate. Initially, L contains two nodes n_s and n_t. Therefore, we have only one choice to insert the node between n_s and n_t. After the first node is inserted into L, there are two (C_1^2) choices for inserting the second module into L. Similarly, there are at most C_1^i choices for inserting the ith node into L if no node is deleted from L. Therefore, the solution space of the CS is bounded by

$$m!1C_1^2C_1^3,\ldots,C_1^m = (m!)^2.$$

Thus, the theorem follows. ■

It should be noted that, in addition to the number of modules, the solution space of the CS also depends on the dimensions of the modules. The above theorem considers the worst case for the CS—all modules appear in the contour all the time during packing. Obviously, it is quite often that only part of the modules are in the contour. Therefore, the practical solution space of the CS is significantly smaller than $(m!)^2$.

Theorem 3: There exists a placement corresponding to each CS.

Proof: The CS uses a packing sequence S of modules, as well as the corresponding bends D of the modules to describe a compacted placement. There exists a unique placement corresponding to a CS if each module is placed at the designated bend, as defined in the CS. If the designated bend is not available, there always exists a bend associated with the current contour for placing the module. ■

Theorem 4: The DSP packing scheme packs modules correctly in $O(m)$ time, where m is the number of modules.

Proof: The DSP scheme packs modules correctly if it can place modules at the designated bends defined in a CS. For each term $(b_i, [b_j, b_k])$ in the CS, it inserts a node n_i between two neighboring nodes n_j and n_k in L, which corresponds to placing module b_i at the bend $[b_j, b_k]$. Since $n_j(n_k)$ corresponds to the module $b_j(b_k)$ left to (below) b_i, the $x(y)$ coordinate of b_i is $x'_j(y'_k)$. Since a contour is formed by modules that are not dominated by others in the x and y directions, those modules that are dominated by b_i must be removed after b_i is placed in order to place the next module correctly. The predecessor (successor) n_p's of n_i in L are deleted during DSP if y'_p's (x'_p's) are smaller than $y'_i(x'_i)$. It is clear that x'_p's (y'_p's) are smaller than $x_i(y_i)$ for those nodes n_p's before (behind) n_i. Therefore, b_p's are dominated by b_i if their $y(x)$ coordinates y'_p's (x'_p's) are also smaller than $y'_i(x'_i)$. The process is repeated until no module is available. Therefore, DSP packs modules correctly for a CS.

The DSP takes a constant time to insert a node n_i into the designated location of L if we keep a pointer for each node in L. If there are m modules in a CS, the time complexity to insert nodes into L is $O(m)$. In addition to inserting nodes into L, we also have to delete those nodes that are dominated by the inserting nodes. Since a node can be deleted at most one time, the cost of deleted nodes is no more than $O(m)$. Therefore, the time complexity of the DSP scheme is $O(m)$ time. ■

Based on the above discussion, we have the following theorem:

Theorem 5: The CS is a P-admissible representation.

Proof: To prove the CS is a P-admissible representation, we have to show that it satisfies the four conditions proposed by Murata *et al.* [8] as follows:

1) By Theorem 2, its solution space is $O((m!)^2)$ and, thus, finite.

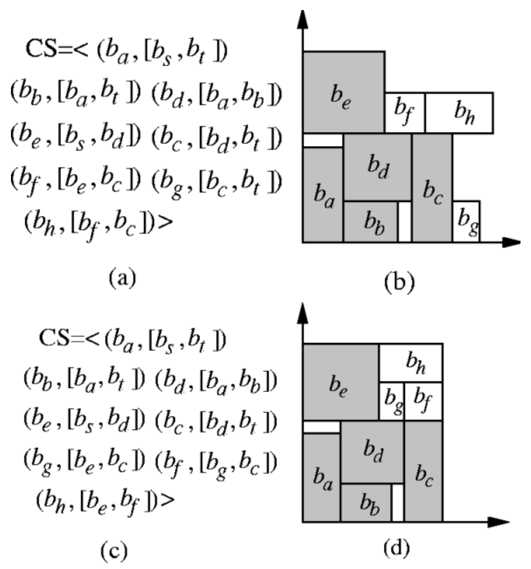
CS=< (b_a, [b_s, b_t]) (b_b, [b_a, b_t]) (b_d, [b_a, b_b]) (b_e, [b_s, b_d]) (b_c, [b_d, b_t]) (b_f, [b_e, b_c]) (b_g, [b_c, b_t]) (b_h, [b_f, b_c])>
(a)
(b)
CS=< (b_a, [b_s, b_t]) (b_b, [b_a, b_t]) (b_d, [b_a, b_b]) (b_e, [b_s, b_d]) (b_c, [b_d, b_t]) (b_g, [b_e, b_c]) (b_f, [b_g, b_c]) (b_h, [b_e, b_f])>
(c)
(d)

Fig. 5. Incremental update for cost evaluation. (a) and (b) [(c) and (d)] show a CS and the corresponding placement. Note that the CS's in (a) and (c) are the same as in the first five terms; the coordinates of the modules remain the same in (c) and (d).

2) By Theorem 3, it guarantees a feasible packing for each CS.
3) By Theorem 4, packing and cost evaluation can be performed in $O(m)$ time.
4) By Theorem 1, all compacted placements can be represented by CS and, thus, the optimal solution exists in the CS solution space. Further, the best evaluated packing in the CS solution space corresponds to an optimal placement.

■

C. *Incremental Update During Packing*

By the packing scheme proposed in the above section, we only need to recompute the coordinates of modules after the lth term if the new CS has the same first l terms as the original CS. Fig. 5(a) and (b) [(c) and (d)] shows a CS and the corresponding placement. Note that the CS of Fig. 5(a) differs from that of Fig. 5(c) in only the last three terms. Therefore, if the coordinates of the modules in the CS of Fig. 5(a) have been obtained, only the coordinates of modules b_f, b_g, and b_h in these terms need to be recomputed [i.e., the coordinates of b_a, b_b, b_c, b_d, and b_e remain the same, as in Figs. 5(b) and (d)]. However, to facilitate such an incremental update, we need to know L for the placement $\mathcal{P}'$ of modules in the first l terms to continue the packing scheme.

Traditional simulated annealing-based floorplan design algorithms perturb a given solution, and then compute coordinates of modules from scratch in each perturbation. Different from these methods, we first determine those modules in the first l terms that do not change before the perturbation and record the L for the placement P' of these modules. Thus, the packing of the new solution can go on from the $(l+1)$th module.

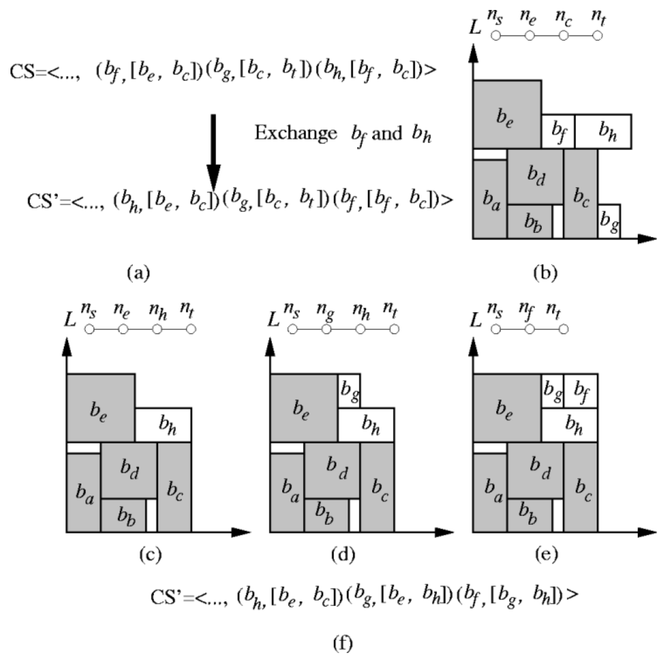
CS=<..., (b_f, [b_e, b_c])(b_g, [b_c, b_t])(b_h, [b_f, b_c])>
Exchange b_f and b_h
CS'=<..., (b_h, [b_e, b_c])(b_g, [b_c, b_t])(b_f, [b_f, b_c])>
(a)
(b)
(c)
(d)
(e)
CS'=<..., (b_h, [b_e, b_c])(b_g, [b_e, b_h])(b_f, [b_g, b_h])>
(f)

Fig. 6. Example of exchanging two modules b_f and b_h in S_6 and S_8 for the CS in Fig. 5(a). (a) CS after the modules in S_6 and S_8 have been exchanged. (b) L for those modules b_a, b_b, b_c, b_d, and b_e, whose coordinates remain the same. (c)–(e) Resulting placement and L after the modules b_h, b_g, and b_f have been packed, respectively. (f) Resulting CS after the operation.

TABLE I
FIVE MCNC BENCHMARK CIRCUITS

Circuit	# of modules	# of pads	# of nets	# pins
apte	9	73	97	214
xerox	10	107	203	696
hp	11	43	83	264
ami33	33	42	123	480
ami49	49	24	408	931

IV. SOLUTION PERTURBATION

We develop a simulated annealing-based algorithm [5] by using the CS for nonslicing floorplan design. Given an initial solution represented by a CS, the algorithm perturbs the CS to obtain a new CS.

The simulated annealing algorithm is described as follows. We randomize an initial CS. The corresponding placement of the CS can be obtained by the DSP scheme. We then perturb a CS into another CS to search for a better solution. As mentioned earlier, we can obtain a placement incrementally after each perturbation. We apply the following four perturbations to obtain a new CS.

- *Exchange*: Exchange two modules in S_i and S_j.
- *Insert*: Insert the ith term between the jth and $(j+1)$th terms.
- *Rotate*: Rotate the module in S_i.
- *Randomize*: Randomize a new D_i for the module in S_i by choosing arbitrary neighboring nodes in L.

For the exchange and insert (rotate and randomize) operations, the first l terms of the given CS will not be changed during perturbation, where $l = \min\{i, j\} - 1 (l = i - 1)$. Therefore, for each perturbation, we only need to consider the modules

TABLE II
AREA AND RUN-TIME COMPARISONS AMONG O-TREE (SUN SPARC ULTRA60), B*-TREE (ON SUN SPARC ULTRA-I), ENHANCED O-TREE (ON SUN SPARC ULTRA60), CBL (ON SUN SPARC 20), TCG (ON SUN SPARC ULTRA60), AND CS (ON SUN SPARC ULTRA60) FOR AREA OPTIMIZATION. (*CBL MIGHT NOT USE THE SAME HP CIRCUIT SINCE IT REPORTS AN AREA OF ABOUT SEVEN TIMES LARGER THAN OTHERS)

Circuit	# of modules	O-tree [3]		B*-tree [1]		enhanced O-tree [12]		CBL [4]		TCG [7]		CS	
		Area mm^2	Time sec	Area mm^2	Time sec	Area mm^2	Time sec	Area mm^2	Time sec	Area mm^2	Time sec	Area mm^2	Time sec
apte	9	47.1	38	**46.92**	7	**46.92**	11	NA	NA	**46.92**	1	**46.92**	1
xerox	10	20.1	118	**19.83**	25	20.21	38	20.96	30	**19.83**	18	**19.83**	54
hp	11	9.21	57	**8.947**	55	9.16	19	*	*	**8.947**	20	**8.947**	6
ami33	33	1.25	1430	1.27	3417	1.24	118	1.20	36	1.20	306	**1.18**	530
ami49	49	37.6	7428	36.80	4752	37.73	406	38.58	65	36.77	434	**36.28**	851

after the lth term and perform an incremental update on the existing packing (solution). The coordinate of module b_i in S_i, $i = l+1, \ldots, m$ can be obtained by inserting a node n_i into two neighboring nodes n_j and n_k in L if $D_i = [b_j, b_k]$. However, if the designated nodes do not exist in L, we randomly insert the node n_i into two arbitrary neighboring nodes n_q and n_r in L and, thus, $D_i = [b_q, b_r]$. Note that we can guarantee a feasible solution after each perturbation by applying this process.

Fig. 6 illustrates the procedure to perturb the CS shown in Fig. 5(a) using the exchange operation. If two modules b_f and b_h in S_6 and S_8 are exchanged, we have the new CS shown in Fig. 6(a). Fig. 6(b) shows the placement and L for the CS before perturbation. Modules b_a, b_b, b_c, b_d, and b_e are in the first five terms of the CS, and will not be changed for this perturbation since $l = \min\{6, 8\} - 1 = 5$ here. The coordinates of the modules in the last three terms of the CS can be obtained by their corresponding bends. (We insert nodes between two designated neighboring nodes according to their bends.) Fig. 6(c) shows the resulting placement and L after we insert the node n_h between the nodes n_e and n_c in the L of Fig. 6(b). For b_g, we then cannot place it at the designated bend $[b_c, b_t]$ because there do not exist two adjacent nodes n_c and n_t in the L of Fig. 6(c). Therefore, we randomly insert n_g into two arbitrary neighboring nodes in L. There are three candidate bends for placing b_g : $[b_s, b_e]$, $[b_e, b_h]$, and $[b_h, b_t]$ (see L and the placement). If we insert n_g between n_e and n_h (the new bend of b_g becomes $[b_e, b_h]$), the resulting placement and L is given in Fig. 6(d). Similarly, we intend to insert n_f between nodes n_f and n_c for the module b_f in the L of Fig. 6(d). However, there do not exist two neighboring nodes n_f and n_c in the L of Fig. 6(d); thus, we randomly insert it between the nodes n_g and n_h [see Fig. 6(e) for the resulting placement and L]. Finally, we have the resulting CS shown in Fig. 6(f).

V. EXPERIMENTAL RESULTS

Based on simulated annealing [5], we implemented the CS representation in the C++ programming language on a 433-MHz SUN Sparc Ultra-60 workstation with 1-GB memory. We compared the CS with the O-tree [3], B*-tree [1], enhanced O-tree [12], CBL [4], and TCG [7], which were recently published, based on the five MCNC benchmark circuits listed in Table I. Columns 2–5 in Table I list the respective numbers of modules, I/O pads, nets, and pins of the five circuits in the circuits.

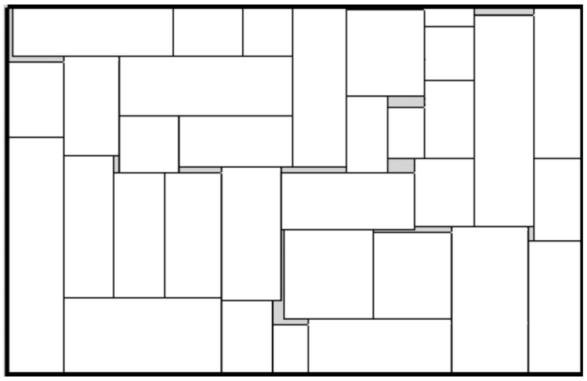
Fig. 7. Resulting placement of ami33 for optimizing area alone (area $= 1.178$ mm^2).

TABLE III
WIRELENGTH AND RUN-TIME COMPARISONS AMONG O-TREE (ON SUN SPARC ULTRA60), ENHANCED O-TREE (ON SUN SPARC ULTRA60), TCG (ON SUN SPARC ULTRA60), AND CS (ON SUN SPARC ULTRA60) FOR WIRELENGTH OPTIMIZATION

Circuit	O-tree		enhanced O-tree		TCG		CS	
	Wire mm	Time sec	Wire mm	Time sec	Wire mm	Time sec	Wire mm	Time sec
apte	**317**	47	**317**	15	363	2	348	23
xerox	368	160	372	39	366	15	**329**	13
hp	153	90	150	19	143	10	**142**	8
ami33	52	2251	52	177	**44**	52	**44**	470
ami49	636	14112	629	688	**604**	767	625	1268

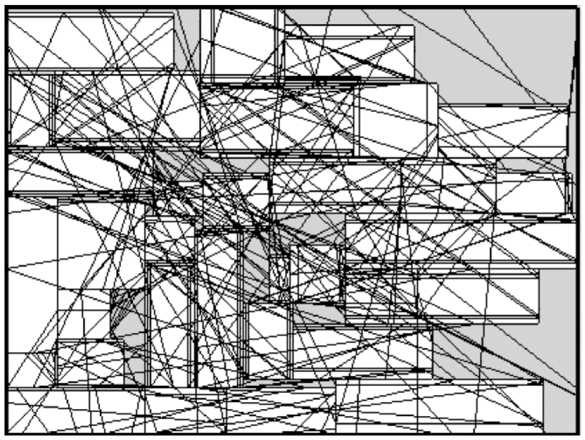
Fig. 8. Resulting placement of ami33 for optimizing wire alone (wire $= 43.67$ mm).

TABLE IV
AREA, WIRELENGTH, AND RUN-TIME COMPARISONS AMONG O-TREE (ON SUN ULTRA60), ENHANCED O-TREE (ON SUN ULTRA60), CBL (ON SUN SPARC 20), TCG (ON SUN ULTRA60), AND CS (ON SUN SPARC ULTRA60) FOR SIMULTANEOUS AREA AND WIRELENGTH OPTIMIZATION

Circuit	O-tree			enhanced O-tree			CBL			TCG			CS		
	Area mm^2	Wire mm	Time sec	Area mm^2	Wire mm	Time sec	Area mm^2	Wire mm	Time sec	Area mm^2	Wire mm	Time sec	Area mm^2	Wire mm	Time sec
apte	51.9	321	47	52.0	321	14	-	-	NA	48.5	378	50	48.5	380	29
xerox	20.4	381	142	20.4	381	41	20.2	404	NA	20.4	385	114	20.4	381	40
hp	9.5	153	84	9.4	152	21	NA	NA	NA	9.5	152	59	9.6	149	27
ami33	1.28	51	2349	1.30	52	205	1.23	52	NA	1.24	50	939	1.25	48.1	476
ami49	39.6	689	15318	39.9	703	700	38.4	733	NA	38.2	663	3613	38.2	690	2103

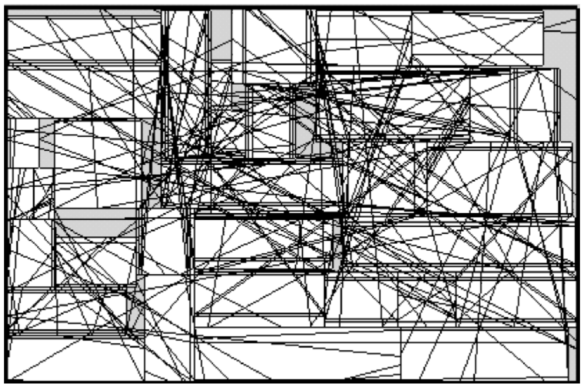
Fig. 9. Resulting placement of ami33 for simultaneous optimizing area and wire (area $= 1.254\ \mathrm{mm}^2$ and wire $= 48.13\ \mathrm{mm}$).

Our experiments consist of three parts, i.e., area, wirelength, and simultaneous area and wirelength optimizations. The area of a placement is measured by that of the minimum bounding box enclosing the placement. The area and run-time comparisons among the O-tree, B*-tree, enhanced O-tree, CBL, and TCG are listed in Table II. As shown in Table II, the CS achieves the best area utilization among the previous works for all of the circuits. Fig. 7 shows the resulting placement for ami33 for area optimization alone.

For timing optimization, we estimated the wirelength of a net by half of the perimeter of the minimum bounding box enclosing the net. The wirelength of a placement is given by the summation of the wirelengths of all nets. The comparisons with the O-tree, enhanced O-tree, and TCG are listed in Table III. The results show that the CS achieves the best wirelength compared with previous works. (Note that there is not a comparison with the B*-tree and CBL here since they did not report the results on wirelength.) Fig. 8 shows the resulting placement for ami33 for wirelength optimization alone $(\mathrm{wire} = 43.67\ \mathrm{mm})$.

For simultaneous timing and wire optimization, we give the same weights for area and wirelength in the cost function. The area and wire comparisons among the O-tree, enhanced O-tree, CBL, and TCG are listed in Table IV, and our results are comparable to the best published results. (Note that there is not a comparison with the B*-tree here since they did not report the results on simultaneous area and wirelength optimization.) Fig. 9 shows the resulting placement for ami33 for simultaneous area and wirelength optimization (area $= 1.254\ \mathrm{mm}^2$ and wire $= 48.13\ \mathrm{mm}$).

VI. CONCLUDING REMARKS

We have presented the CS representation for nonslicing floorplans. The CS is P-admissible. It is very simple and can be implemented easily. Cost evaluation can also be performed incrementally on the CS. In particular, it induces a generic worst case linear-time packing scheme that can also be applied to other existing representations. Experimental results have shown that the CS is a very promising representation.

REFERENCES

[1] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for nonslicing floorplans," in *Proc. ACM/IEEE Design Automation Conf.*, 2000, pp. 458–463.

[2] J. Cong, T. Kong, and D. Pang, "Buffer block planning for interconnect-driven floorplanning," in *Proc. IEEE/ACM International Computer-Aided Design Conf.*, 1999, pp. 358–361.

[3] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-tree representation of nonslicing floorplan and its applications," in *Proc. ACM/IEEE Design Automation Conf.*, 1999, pp. 268–273.

[4] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, "Corner block list: An effective and efficient topological representation of nonslicing floorplan," in *Proc. IEEE/ACM Int. Computer-Aided Design Conf.*, 2000, pp. 8–12.

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

[6] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston, 1976.

[7] J.-M. Lin and Y.-W. Chang, "TCG: A transitive closure graph-based representation for nonslicing floorplans," in *Proc. ACM/IEEE Design Automation Conf.*, 2001, pp. 764–769.

[8] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing based module placement," in *Proc. IEEE/ACM Int. Computer-Aided Design Conf.*, 1995, pp. 472–479.

[9] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module placement on BSG-structure and IC layout applications," in *Proc. IEEE/ACM Int. Computer-Aided Design Conf.*, 1996, pp. 484–491.

[10] T. Ohtsuki, N. Suzigama, and H. Hawanishi, "An optimization technique for integrated circuit layout design," in *Proc. ICCST*, 1970, pp. 67–68.

[11] R. H. J. M. Otten, "Automatic floorplan design," in *Proc. ACM/IEEE Design Automation Conf.*, 1982, pp. 261–267.

[12] Y. Pang, C. K. Cheng, and T. Yoshimura, "An enhanced perturbing algorithm for floorplan design using the O-tree representation," in *Proc. ACM Int. Physical Design Symp.*, 2000, pp. 168–173.

[13] S. M. Sait and H. Youssef, *VLSI Physical Design Automation: Theory and Practice*, Singapore: World Scientific, 1999.

[14] X. Tang and D. F. Wong, "FAST-SP: A fast algorithm for block placement based on sequence pair," in *Proc. ACM Asia and South Pacific Design Automation Conf.*, 2001, pp. 521–526.

[15] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proc. ACM/IEEE Design Automation Conf.*, 1986, pp. 101–107.

Jai-Ming Lin received the B.S., M.S., and Ph.D. degrees in computer and information science from the National Chiao Tung University, Taiwan, R.O.C., in 1996, 1998, and 2002, respectively.

He is currently a Computer-Aided Design (CAD) Engineer with the Realtek Semiconductor Corporation, Hsinchu, Taiwan, R.O.C. His research interest focuses on physical design.

Yao-Wen Chang (S'94–M'96) received the B.S. degree from the National Taiwan University, Taiwan, R.O.C., in 1988, and the M.S. and the Ph.D. degrees from The University of Texas at Austin, in 1993 and 1996, respectively, all in computer science.

He is currently an Associate Professor with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University. During the summer of 1994, he was with the VLSI Design Group, IBM T. J. Watson Research Center, Yorktown Heights, NY. From 1996 to 2001, he was on the faculty of the Department of Computer and Information Science, National Chiao Tung University, Taiwan, R.O.C. His research interests are physical design automation, architectures, and systems for VLSI and combinatorial optimization.

Dr. Chang is a member of the IEEE Circuits and Systems Society, the Association for Computing Machinery (ACM), and the ACM/Special Interest Group on Design Automaiton (SIGDA). He has served on the Technical Program Committees of several international conferences on VLSI design automation. He was the recipient of the Best Paper Award presented at the 1995 IEEE International Conference on Computer Design (ICCD'95) for his work on field-programmable gate array (FPGA) routing. He received reviewers' Best Paper nominations at the 2000 ACM/IEEE Design Automation Conference (DAC'00) for his research on the B^*-tree floorplan representation and the Best Paper nomination at the 2002 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'02) for his work on multilevel routing. He was the recipient of a 2000 inaugural all-university Excellent Teaching Award presented by the National Chiao Tung University.

Shih-Ping Lin received the B.S. and M.S. degrees in computer and information science from the National Chiao Tung University, Taiwan, R.O.C., in 2000 and 2002, respectively, and is currently working toward the Ph.D. degree in electronics engineering at the National Chiao Tung University.

His research interests include floorplan and routing designs in VLSI.