An Enhanced Q-Sequence Augmented with Empty-Room-Insertion and Parenthesis Trees

Changwen ZHUANG[†] Keishi SAKANUSHI[‡] Liyan JIN[‡] Yoji KAJITANI[†]

[†]Information and Media Sciences
University of Kitakyusyu
1-1, Hibikino, Wakamatsu-ku, Kitakyushu-shi
Fukuoka, 808-0135, Japan
E-mail:{changwen, kajitani}@env.kitakyu-u.ac.jp

[‡]Communications and Integrated Systems
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku
Tokyo, 152-8552, Japan
E-mail:{keishi, lyjin}@lab.ss.titech.ac.jp

Abstract

After the discussion on the difference between floorplanning and packing in VLSI placement design, this paper adapts the floorplanner that is based on the Q-sequence to a packing algorithm. For the purpose, some empty room insertion is required to guarantee not to miss the optimum packing. To increase the performance in packing, a new move that perturbs the floorplan is introduced in terms of the Parenthesis-Tree Pair. A Simulated Anealing based packing search algorithm was implemented. Experimental results showed the effect of empty room insertion.

1 Introduction

With the move towards the deep-submicron era, the circuit design is entering into a new world in which the target system is so large and complicated that hierarchical and IP based design methodologies are dominating, making the placement more and more important.

Placement in VLSI design is to arrange the modules on a chip under the constraint that *no two modules are overlapping* while controlling the area, wire length, and other performance indices to be optimal.

Possible ideas to reinforce this constraint are *packing* and *floorplanning*. These terms have been used confusingly so far as to imply similar meanings. But here we would like to distinguish them as follows. Given n modules, floorplanning consists of three stages of *partition* of the chip area into n rooms, *one-to-one embedding* of modules into rooms, and *1-dimensional compaction*. While packing gives every pair of modules a constraint to avoid overlapping. A popular example is that module A is right of B, meaning that A's left border is right of B's right border. And then determine the position of each module.

The difference between floorplanning and packing could be clear if we consider them to use as an alternative of the other.

Take an example of $n = 4$ modules with size $height \times width$ being $A = 3 \times 2$, $B = 2 \times 3$, $C = 2 \times 4$, and $\mathcal{D} = 3 \times 3$. The packing problem is to find a placement whose bounding box area (BBA) is minimum. The solution is shown in Fig.1(a). This placement is not obtained by the conventional use of the $n = 4$ floorplanner whatever it is. This is obtained by embedding 4 modules into 5 rooms. So we have to prepare $5 = n + 1$ room floorplan allowing one to be empty. If we were to stick to $n = 4$ room floorplan, the packing shown in (b) would be output as the optimal.

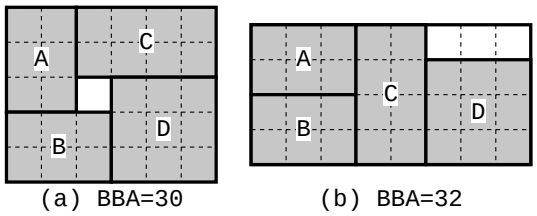


Figure 1: (a) A minimum area packing not obtainable by any $n = 4$ floorplanner. (b) A minimum area packing obtainable by $n = 4$ floorplanner

On the other hand, suppose we use a packing algorithm to construct a floorplan of $n = 4$ rooms. Given a packing of 4 modules, we draw horizontal and vertical line segments so that each module is enclosed. The resultant rectangle areas are rooms. If the placement by a packing algorithm is the one as shown in Fig.1(a), we will have a floorplan of $n \geq 5$ rooms, failed to produce a floorplan of 4 rooms.

Thus, floorplanning and packing cannot be alternatives for others without some careful adaptation. They may say that practically the difference in performance, e.g. in area, would be negligible. But recent experiments for a benchmark ami49 showed a significant difference according to the number of rooms that allow to be empty. See [14].

In the following, a brief history of development of packing and floorplanning is reviewed to describe our motivation.

1.1 Packing

Before the invention of the BSG [9], automated packing had been successful only for the *slicing placement* which is characterized by a recursive cut of the rectangle area. A representation in terms of the *slicing tree* [10] has been attractive to practical applications and its string expression, *normalized polish expression*[15], accelerated this trend. These successes encouraged trials to generalize the procedure slic-

ing to zig-zag cutting that leads to a certain success, at least theoretically. (One reference is cited [3].)

In 1994, a grid data structure by the name of *Bounded-Sliceline-Grid (BSG)*[9] came out. It is the first data structure that can represent any placement, and it was followed by the *Sequence-Pair*[7]. Their structures are suited for stochastic search and implementations based on Simulated Annealing show a good performance in minimizing the area.

Guo et. al. [4] and T. Takahashi[13] observed that the *rooted ordered-tree* corresponds to the area of a packing after 1-Dimensional compaction. Thus, they came to the data structure, *Ordered-Tree (O-tree)*. All these tools (BSG, Sequence-Pair, O-Tree) show a certain potential in packing with hundreds modules and practical hours of simulated annealing. Therefore, we may conclude that the packing problem of rectangles has been solved.

1.2 Floorplanning

In a floorplan, two rooms are said *channel-adjacent* to each other if they share one segment as the border. In our model, each segment in a floorplan represents a channel. The basic objective of floorplanning is to satisfy a certain channel-adjacency request among specified rooms and boundaries of the chip. Before the invent of the Q-seq[12] and CBL[5], there has been no significantly useful tools for optimal floorplanning except slicing floorplan. So efforts[8, 6] have been devoted to adapt the Sequence-Pair to be a tool for floorplanning. The problem they tackled was to provide a way to generate only the Sequence-Pairs that correspond to the placements that lead floorplans without empty rooms.

In 2000, the Q-seq[12] by Sakanushi et.al. and Corner-Block-List (CBL)[5] by Hong et. al. were proposed. They were independent though the principle was revealed the same as is found in the contribution in 1930 by Abe[1].

1.3 Floorplanning to Packing

The new floorplanning algorithms Q-seq and CBL are very much satisfactory in stochastic search, especially with respect to the channel-adjacency. Thus, they lost the meaning to adapt the packing algorithm to floorplanning.

Conversely, adaptation of floorplanning algorithm for packing began to have a meaning since these new algorithms are very quick in transformation of solutions. Moreover, as an application to VLSI layout design, floorplanning is more suited than packing to the VLSI circuit design since a module is not merely a box but has various peripherals and routing space around and floorplanning secures a space for each module.

Still floorplanning is required to be applicable to packing. For the purpose, we have to allow the existence of *empty rooms*, rooms that contain no modules. Then the problem to be solved first of all is to estimate the number of empty rooms to be provided not to miss the minimum area packing. For this problem, Murata et. al. [8] showed that the upper bound is $n + \lceil (n-2)/2 \rceil \lfloor (n-2)/2 \rfloor$. Later Kodama et. al.[6] improved this to $n-3$ with a conjecture. In this paper, we show more tight upper bound which is equal to the value Kodama et. al. had conjectured.

Another contribution is the introduction of a new move in addition to the previously introduced moves[12]. It is described on the *parenthesis tree-pair* representation in a very simple form that would need a complicated description otherwise. This move is global and complementary to other moves so that the diameter of the solution space is linear to n.

Experiments on MCNC benchmarks showed that we can get better solutions than others in shorter time.

The rest of the paper is organized as follows. Section 2 introduces the Q-sequence briefly. Section 3 is devoted to the study of floorplans with empty rooms that are redundant for packing. Section 4 provides the parenthesis trees and operations on them. Section 5 presents a new move procedure and proof of the reachability of the solution space. Section 6 is for experiments and analysis. Section 7 concludes the paper.

2 Q-sequence

We introduce the original Q-seq briefly here.

2.1 Encoding

We first define several terms. In a floorplan, segments dissecting the chip into rooms are called *segs*, which always end at one seg or boundary in T-form. Given a room i, the seg that ends at the room i's right-bottom corner is the *prime seg* of room i. If the prime seg, s, is vertical (horizontal), the rooms located on the right-hand (below) side and adjacent to s are called *associated rooms*, and the highest (most left) one is the *next room* of room i. For each room i, label $\mathcal{R}_i$ and $\mathcal{B}_i$ are called *positional symbols*, so there are two types of positional symbols.

We can encode a Q-Seq, Q, from a floorplan as follows: Let $i = 1$, and the current room be the room that locates on the left-top corner of the chip, and Q is null. We record i into Q, and label the current room as room i. If the prime seg of room i is vertical (horizontal), from bottom (right) to top (left), for each associated room j, we record a positional symbol $\mathcal{R}_j(\mathcal{B}_j)$ into Q.

Let the current room be the next room of room i, and $i = i + 1$, then iterate until room i has no associated room. For example in Fig.2, after iteration, Q becomes

$$1\mathcal{R}_3\mathcal{R}_2 2\mathcal{B}_6\mathcal{B}_4\mathcal{B}_3 3\mathcal{R}_4 4\mathcal{B}_5 5\mathcal{R}_6 6.$$

All rooms in the floorplan have been labeled in Abe-order[1]. We can get other strings denoted by L and T by the following ways. From bottom to top, for each room i located on the left boundary of the chip, we record a positional symbol $\mathcal{R}_i$ into L. In Fig.2, L becomes $\mathcal{R}_5\mathcal{R}_1$. From right to left, for each room j located on the top boundary of the chip, we record a positional symbol $\mathcal{B}_j$ into T. T becomes $\mathcal{B}_2\mathcal{B}_1$.

Finally, we insert L and T at the head of above Q, and Q becomes

$$\mathcal{R}_5\mathcal{R}_1\mathcal{B}_2\mathcal{B}_1 1\mathcal{R}_3\mathcal{R}_2 2\mathcal{B}_6\mathcal{B}_4\mathcal{B}_3 3\mathcal{R}_4 4\mathcal{B}_5 5\mathcal{R}_6 6 \qquad (1)$$

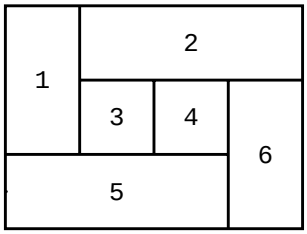


Figure 2: A floorplan with 6 rooms

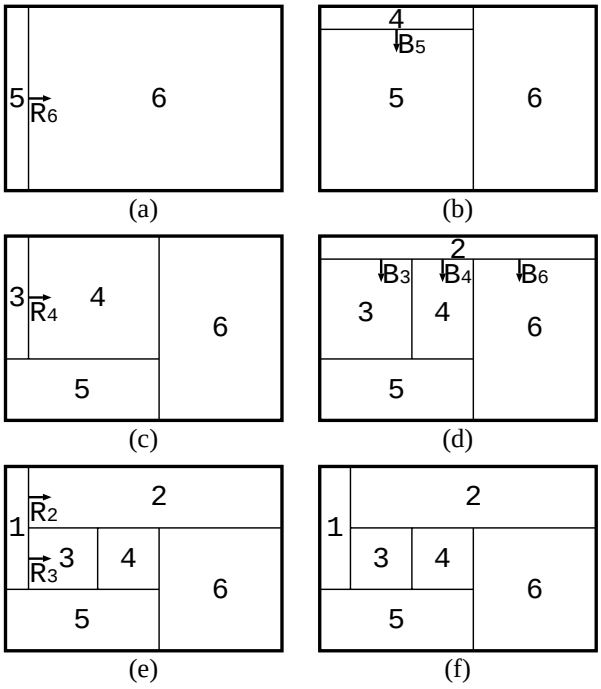


Figure 3: Decoding of the Q-seq of Eq.(1)

which is the final Q-seq corresponding to the floorplan in Fig.2.

Let *interval* $i (0 < i < n)$, denoted by $I(i)$, be the subsequence between room i and $i+1$ in a Q-Seq, not including room labels, and $|I(i)|$ be the number of positional symbols in $I(i)$. All positional symbols in $I(i)$ are the same type, and $|I(i)| > 0$ where $0 < i < n$.

2.2 Decoding

The following is the framework of the decoding procedure of the Q-seq and decoding of Eq.(1) is shown in Fig.3.

— Decoding Procedure —
Input: A Q-seq with module assignment, and module sizes.
Output: Placement of modules

Initial Step: Prepare a rectangular chip where there is only one room labeled by n.

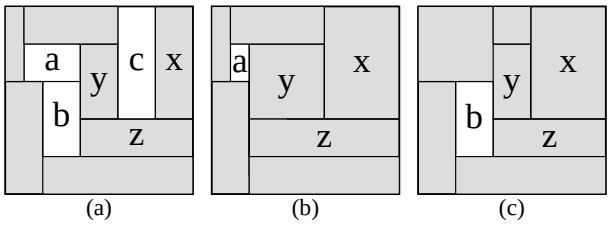


Figure 4: (a) Redundant floorplan, and equivalent one w.r.t. area (b) or (c)

Main Step: Repeat the following steps from $i = n-1$ to $i = 1$.

If $I(i)$ consists of $\mathcal{R}$'s: Insert room i from the left of the chip after pushing aside top $|I(i)|$ rooms that are adjacent to the left boundary of the chip. Note that room x whose $\mathcal{R}_x$ is in $I(i)$ is pushed. (See Fig. 3(a), (c), and (e).)

If $I(i)$ consists of $\mathcal{B}$'s: Insert room i from the top of the chip analogously. (See Fig.3(b), and (d).)

— End of Decoding Procedure —

Both encoding a floorplan to the unique Q-seq and decoding a Q-seq to the unique floorplan need $O(n)$ time.

3 Empty Room Insertion

The Q-seq is enhanced to equip with additional rooms as empty rooms. A room containing a module is called a *solid room*. The number of added rooms shall be large enough so that any packing is obtained through floorplanning. For example, the optimal solution shown in Fig.1(a) was obtained if the floorplanner had been enhanced to include one or more additional rooms.

Given a floorplan. An example in Fig.4(a) where three empty rooms a, b, and c are observed. Different from the modules in packing, a room is merely a dissected region of the chip. Hence an empty room like c has not a meaning to exist in this situation since solid room x can expand to merge c, keeping its shape being rectangular. In other words, from the floorplan thus modified, we will get the same packing after the 1D compaction.

Consider the situation more in detail. Let an end point of a seg be called a *tip* of the seg. Note that a seg has two tips. Then, the above observation is restated as "there is a seg whose both tips belong to the same empty room".

Also we observe that two empty rooms a and b have not a meaning since either one of floorplans in Fig.4(b) or (c) attains a packing with area not exceeding that by the floorplan in (a). This situation is stated as "there is a seg one of whose tips belongs to two empty rooms".

Thus, the floorplan in Fig.4(a) is redundant for the area packing with respect to the number of empty rooms. This is

replaced with ones in Fig.4(b) and (c) with less number of empty rooms.

Our formal definition is as follows.

> **Redundant floorplan:** A seg is called *reducible* if its both tips belong to one room or one of its tips belongs to two empty rooms. A floorplan is said *redundant* if it contains reducible segs.

Since our current subject is packing, it is concluded that we have only to search the non-redundant floorplans. In other words, we have only to equip the Q-seq with M empty rooms where M is the maximum number of empty rooms contained in the non-redundant floorplans. Note that M is the function of n and that the length of the enhanced Q-seq is $3(n+M)$. The problem in this section is to determine M.

Any integer n has uniquely a form of

$$n = k^2 + j,\ 0 \le j \le 2k. \tag{2}$$

Theorem 1 *Given n modules, in Eq.(2),*

$$M = \begin{cases} (k-1)^2 & , if \quad 0 \le j \le 1 \\ (k-1)^2 + j - 1\ , if & 2 \le j \le k \\ (k-1)^2 + j - 2\ , if\ k+1 & \le j \le 2k \end{cases} \tag{3}$$

□

Sketch of the proof: For a given non-redundant floorplan, assume it a planar graph, letting segs and intersections be edges and vertices, respectively. Neglecting the outer domain, draw the dual graph putting vertices inside the rooms and connecting two vertices of the adjacent rooms. A vertex corresponding to an empty room is surrounded by a cycle of length 4 or less and whose vertices correspond to the solid rooms and length is 4 or more.

Let the graph obtained from this by deleting the vertices corresponding to the empty rooms be called the *dual solid room graph*. A floorplan and its dual solid room graph are shown in Fig.5(a) and (b), respectively. The number of vertices is n. Hence the problem is reduced to finding a planar graph of n vertices that maximize the number of faces. If $n = k^2$, it is trivial that a solution is the one whose dual solid room graph is the $k \times k$ grid graph. Other cases are spirally constructed at two peripherals up to $2k$ rooms. This construction rule is guessed from the numbering of the vertices in Fig.5. A detailed discussion leads to the theorem. □

If we represent Eq.(2) in one equation, it is

$$M = n - \lfloor \sqrt{4n-1} \rfloor$$

which is identical to Kodama's Conjecture in [6].

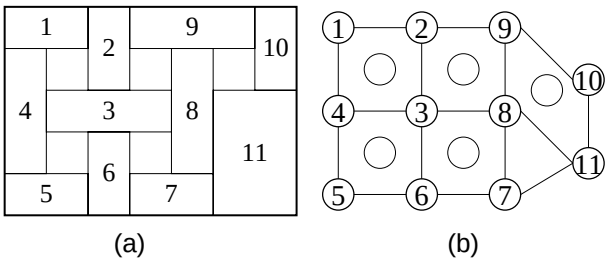


Figure 5: A non-redundant floorplan and its dual solid room graph construction when $n = 11 = 3^2 + 2, M = (3-1)^2 + 2 - 1 = 5$.

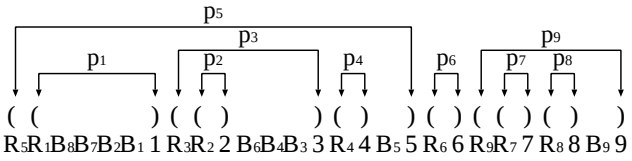


Figure 6: Q-seq and its $\mathcal{R}$ parenthesis system

4 Parenthesis Constraint Trees

4.1 Parenthesis System

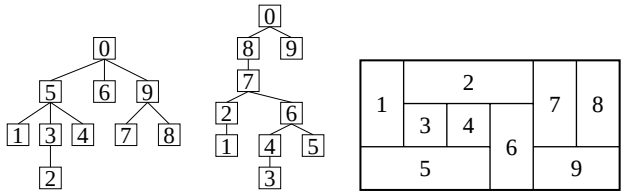
The Q-seq is a string including n room labels ordered by Abe-order[12] and $2n$ positional symbols, which are classified into two types, $\mathcal{R}$ type and $\mathcal{B}$ type, respectively, where n is the total number of solid rooms and empty rooms.

In the rest of the paper, a room can be either a solid room or an empty room and n is the total number of solid room and empty rooms, if no special declaration.

We first consider the $\mathcal{R}$ positional symbols and room labels in a Q-seq. Fig.6 shows a Q-seq for a packing with 9 rooms, and we put a left parenthesis "(" above $\mathcal{R}_i$, and a right parenthesis ")" above the room label i, where i is 1, 2, . . ., n. Let p_i denote the pairing of $\mathcal{R}_i$ and i, then there are n $\mathcal{R}$ pairings, p_1, p_2, . . ., p_n, respectively. For two parings p_i and p_j, p_i is said to be included by p_j, denoted by $p_i \subset p_j$, if p_j includes p_i. For two parings p_i and p_j, p_i is said to be parallel to p_j, denoted by $p_i \parallel p_j$, if p_i does not include p_j and p_j does not include p_i.

For example, in Fig.6, $p_1 \subset p_5$, and $p_1 \parallel p_3$. Any two $\mathcal{R}$ pairings must bear one of the two relations: including relation and parallel relation. No two $\mathcal{R}$ pairings cross each other[12]. Any Q-seq must satisfy $\mathcal{R}$ pairings relation constraints. Similarly, we can get n $\mathcal{B}$ pairings of $\mathcal{B}_i$ and i and $\mathcal{B}$ pairing relation constraints that any Q-seqs must satisfy. $\mathcal{R}$ and $\mathcal{B}$ pairings relation constraints are also called the parenthesis constraint.

Since the Q-seq can represent any packing, we can perturb a packing by modifying the Q-Seq itself. How do we modify the Q-seq efficiently without violating the parenthesis constraint? We utilize two trees, $\mathcal{R}$ parenthesis tree and $\mathcal{B}$ parenthesis tree (simply $\mathcal{R}$-tree and $\mathcal{B}$-tree, respectively),



(a) $\mathcal{R}$-tree (b) $\mathcal{B}$-tree (c) Packing

Figure 7: Parenthesis trees and packing of Q-seq in Fig.6

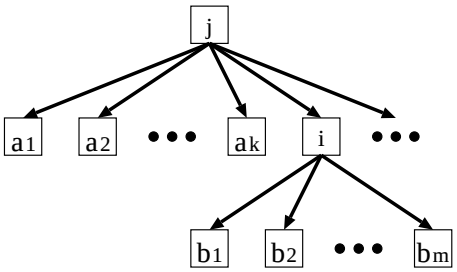


Figure 8: Node i's sibling, children

the structures to represent the relations of n $\mathcal{R}$ pairings and n $\mathcal{B}$ pairings in the Q-seq respectively. We will show that the operations on the trees are very simple while keeping the parenthesis constraints.

4.2 Parenthesis Trees

An $\mathcal{R}$ parenthesis tree is derived from a Q-seq as follows. Each node i in the tree represents an $\mathcal{R}$ pairing p_i, and the root of the tree represents an $\mathcal{R}$ pairing including the whole Q-seq, denoted by p_0. Given a node i, its children correspond to $\mathcal{R}$ pairings that are not included by other pairings in the substring from $\mathcal{R}_i$ to i in the Q-seq except p_i itself, and the children are placed in the same order as in the Q-seq. For example in Fig.6, under p_5, three $\mathcal{R}$ pairings, p_1, p_3, and p_4, are not included by other pairings in the substring from $\mathcal{R}_5$ to 5 in the Q-seq except p_5. Thus node 5 has three children, denoted by node 1, 3 and 4, respectively in Fig.7(a). By recursively using this way, from node 0, we can construct the whole $\mathcal{R}$ parenthesis tree until no leaves including any $\mathcal{R}$ pairings. The $\mathcal{R}$ parenthesis tree of the Q-seq in Fig.6 is shown in the Fig.7 (a). Similarly, we can construct the $\mathcal{B}$ parenthesis tree for the Q-seq. Fig.7(b) and Fig.7(c) are the $\mathcal{B}$ parenthesis tree and the packing of the Q-seq in Fig.6.

4.3 Properties of Parenthesis Trees

We summarize several features of parenthesis trees here.

Property 1: Given two rooms i and j, corresponding to node i and node j respectively in the $\mathcal{R}$ parenthesis tree, if node i is an ancestor of node j, then room i is below room j in the packing.

For example, in Fig.7(a), node 5 has three children. The corresponding three rooms are room 1, room 3, room 4, and it is clear that room 5 is below them.

Property 2: Given two rooms i and j, corresponding to node i and node j respectively in the $\mathcal{R}$ parenthesis tree, if node i is a left sibling of node j, then room i is on the left to room j in the packing.

For example, in Fig.7(a), node 7 and node 8 have the same parent, so they are siblings. Since node 7 is on the left to node 8, then room 7 is on the left to room 8 in the packing.

Property 3: Given two rooms i and j, corresponding to node i and node j respectively in the $\mathcal{B}$ parenthesis tree, if node i is an ancestor of node j, then room i is on the right to room j in the packing.

For example, in Fig.7(b), node 7 has two children, the corresponding two rooms are room 2, room 6, and it is clear that room 7 is on the right to them.

Property 4: Given two rooms i and j, corresponding to node i and node j respectively in the $\mathcal{B}$ parenthesis tree, if node i is a left sibling of node j, then room i is above room j in the packing.

For example, in Fig.7(b), node 4 and node 5 have the same parent, so they are siblings. Since node 4 is on the left of node 5, then room 4 is above room 5 in the packing.

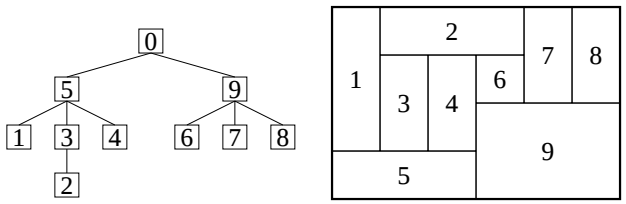
Theorem 2 *The rooms corresponding to the nodes whose parents are the root in the $\mathcal{R}$ parenthesis tree are placed on the bottom boundary in the packing. Analogously, the rooms corresponding to the nodes whose parents are the root in the $\mathcal{B}$ parenthesis tree are placed on the right boundary.*

As shown in Fig.7(c), room 5, 6, and 9 are on the bottom boundary and room 8 and 9 are on the right boundary. Since the Q-seq can find the rooms on the top and left boundaries, we know that the Q-seq can keep all the information of boundary modules.

4.4 Operations on Parenthesis Trees

A Q-seq can be perturbed by moving positional symbols back and forth in the Q-seq. We first discuss how to move an $\mathcal{R}$ positional symbol, e.g. $\mathcal{R}_i$. While moving $\mathcal{R}_i$, the new sequence must satisfy the parenthesis constraint, which ensures the new sequence is a Q-seq representing a feasible packing. When we move $\mathcal{R}_i$ in the Q-seq, the relations between the $\mathcal{R}$ pairing p_i and other $\mathcal{R}$ pairings will change accordingly, which can be described by changing the $\mathcal{R}$ parenthesis tree. When $\mathcal{R}_i$ is moved forward, p_i will include more $\mathcal{R}$ pairings, and when $\mathcal{R}_i$ is moved backward, p_i will include less $\mathcal{R}$ pairings, where $\mathcal{R}_i$ can not be placed behind of room label i, and can not be placed before $\mathcal{R}_j$ if $p_i \subset p_j$.

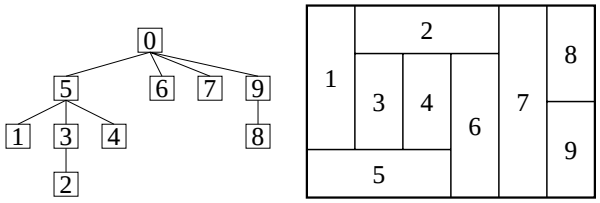
In Fig.8, providing node i has k ($k \geq 0$) left sibling nodes and m ($m \geq 0$) children, $\mathcal{R}_i$ can be moved to just before $\mathcal{R}_a$ ($a = a_1, a_2, \cdots, a_k$) to make paring p_i include more parings corresponding to sibling nodes from a_1 to a_k, and $\mathcal{R}_i$ also can be moved to just before $\mathcal{R}_b$ ($b = b_2, b_3, \cdots, b_m$) to make pairing p_i include less parings. When $\mathcal{R}_i$ is placed just before room label i, p_i includes no pairing, then node i has no children.



(a) $\mathcal{R}$-tree (b) packing

(c) $\mathcal{R}_5\mathcal{R}_1\mathcal{B}_8\mathcal{B}_7\mathcal{B}_2\mathcal{B}_1 1\mathcal{R}_3\mathcal{R}_2 2\mathcal{B}_6\mathcal{B}_4\mathcal{B}_3 3\mathcal{R}_4 4\mathcal{B}_5 5\underline{\mathcal{R}_9}\mathcal{R}_6 6\mathcal{R}_7 7\mathcal{R}_8 8\mathcal{B}_9 9$

Figure 9: Moving $\mathcal{R}_9$ forth in the Q-seq in Fig.6



(a) $\mathcal{R}$-tree (b) packing

(c) $\mathcal{R}_5\mathcal{R}_1\mathcal{B}_8\mathcal{B}_7\mathcal{B}_2\mathcal{B}_1 1\mathcal{R}_3\mathcal{R}_2 2\mathcal{B}_6\mathcal{B}_4\mathcal{B}_3 3\mathcal{R}_4 4\mathcal{B}_5 5\mathcal{R}_6 6\mathcal{R}_7 7\underline{\mathcal{R}_9}\mathcal{R}_8 8\mathcal{B}_9 9$

Figure 10: Moving $\mathcal{R}_9$ back in the Q-seq in Fig.6

In the Q-seq in Fig.6, $\mathcal{R}_9$ can be moved forward and be placed just before $\mathcal{R}_6$, i.e. node 6 becomes child of node 9 in the corresponding $\mathcal{R}$ parenthesis tree (in Fig.7(a)).

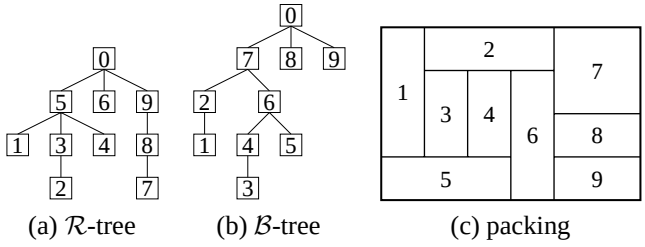
Then the $\mathcal{R}$ parenthesis tree will become the one in Fig.9(a). The corresponding packing and Q-seq are shown in Fig.9(b) and Fig.9(c), respectively.

In the Q-seq in Fig.6, we also can place $\mathcal{R}_9$ just before $\mathcal{R}_8$. Then the corresponding $\mathcal{R}$ parenthesis tree, packing, and Q-seq become the ones shown in Fig.10(a), (b), and (c), respectively. Note that node 7 becomes one of node 9's sibling nodes.

In aforementioned two examples, we just move $\mathcal{R}_i$ to another places, which is very simple. We can move $\mathcal{B}_i$ to-and-fro in the same way. Now, we can summarize two basic operations on the parenthesis trees, adding children denoted by $A(T,i,j)$ and freeing children denoted by $F(T,i,j)$, where T is an $\mathcal{R}$ parenthesis tree or a $\mathcal{B}$ parenthesis tree, i and j are room labels in the Q-seq. $A(T,i,j)$ places $\mathcal{R}_i(\mathcal{B}_i)$ just before $\mathcal{R}_j(\mathcal{B}_j)$ in the Q-seq, then the sibling nodes j and those between node j and i(including node j) become the children of node i in T, where $i > j$. $F(T,i,j)$ move $\mathcal{R}_i(\mathcal{B}_i)$ just before $\mathcal{R}_j(\mathcal{B}_j)$, and all the left siblings of node j become node i's left siblings. $F(T,i,i)$ will free all children of node i, and place $\mathcal{R}_i(\mathcal{B}_i)$ just before room label i in the Q-seq.

Based on $A(T,i,j)$ and $F(T,i,j)$, we can define more two simple operations. Sometime, if there is only $\mathcal{R}_i$ in interval $I(i-1)$ in the Q-seq, we can also move $\mathcal{R}_i$ to elsewhere.

For example, suppose we want in Fig.6 to move $\mathcal{R}_8$ to the position just before $\mathcal{R}_7$. But it left a blank between room 7 and 8 so that $|I(7)| = 0$. At this moment, we should move



(a) $\mathcal{R}$-tree (b) $\mathcal{B}$-tree (c) packing

(d) $\mathcal{R}_5\mathcal{R}_1\mathcal{B}_7\mathcal{B}_2\mathcal{B}_1 1\mathcal{R}_3\mathcal{R}_2 2\mathcal{B}_6\mathcal{B}_4\mathcal{B}_3 3\mathcal{R}_4 4\mathcal{B}_5 5\mathcal{R}_6 6\mathcal{R}_9\underline{\mathcal{R}_8}\mathcal{R}_7 7\underline{\mathcal{B}_8} 8\mathcal{B}_9 9$

Figure 11: The result of moving $\mathcal{R}_8$ in the Q-seq in Fig.6

the $\mathcal{B}_8$ here to fill the blank. After $\mathcal{B}_8$ is moved back and just be placed before room 8, all children of node 8 are freed and become node 8's left siblings in the $\mathcal{B}$ parenthesis tree. The resultant $\mathcal{R}$ parenthesis tree, $\mathcal{B}$ parenthesis tree, packing and Q-seq are shown in Fig.11(a), (b), (c), (d), respectively.

Therefore, we define a new operation $RM(i,j)$ such that i and j are room labels in the Q-seq and $i > j$. $RM(i,j)$ moves $\mathcal{R}_i$ to the position just before $\mathcal{R}_j$ and then place $\mathcal{B}_i$ just before room label i.

In fact, as shown above, $RM(i,j)$ includes two step, first $A(T,i,j)$, where T is the $\mathcal{R}$ parenthesis tree, and then $F(T,i,i)$, where T is the $\mathcal{B}$ parenthesis tree. Similarly, we can define an operation $BM(i,j)$, where i and j are room labels in the Q-seq and $i > j$. $BM(i,j)$ will move $\mathcal{B}_i$ to the position just before $\mathcal{B}_j$ and then place $\mathcal{R}_i$ just before room label i. $BM(i,j)$ includes two step, first $A(T,i,j)$, where T is the $\mathcal{B}$ parenthesis tree, and then $F(T,i,i)$, where T is the $\mathcal{R}$ parenthesis tree. By far, we have introduced four operations on parenthesis trees, $A(T,i,j), F(T,i,i), RM(i,j)$, and $BM(i,j)$, respectively,

by which we can perturb the Q-seqs quickly and easily.

5 Solution Space and Move

5.1 Move Procedure

When Q-seq is used to tackle packing problem with all modules' dimensions, we apply one of the following four packing perturbation operations to perturb a packing: (1) *rotation*, rotate a module, (2) *swap*, exchange the two modules in two rooms, (3) $\mathcal{R}$ *moving*, moving $\mathcal{R}_i$ randomly and feasibly in the Q-seq, (4) $\mathcal{B}$ *moving*, moving $\mathcal{B}_i$ randomly and feasibly in the Q-seq, where $1 \le i \le n$, and n is the number of the rooms.

For rotation, we just need to exchange the width and height of the module in one randomly selected room that is not an empty room.

For swap, we only exchange the module number in two randomly selected rooms, where there is at least one solid room.

For $\mathcal{R}$ moving, if a selected node i has children or sibling nodes in the $\mathcal{R}$ parenthesis tree, we can use $A(T,i,j)$ to add some children or $F(T,i,j)$ to free some children. If $I(i-1)$ in the Q-seq includes only one positional symbol $\mathcal{R}_i$, we can

move $\mathcal{R}_i$ by using $RM(i,j)$.

For $\mathcal{B}$ moving, similarly, if a selected node i has children or sibling nodes in the $\mathcal{B}$ parenthesis tree, we can use $A(T,i,j)$ to add some children or $F(T,i,j)$ to free some children. If $I(i-1)$ in the Q-seq includes only one positional symbol $\mathcal{B}_i$, we can move $\mathcal{B}_i$ by using $BM(i,j)$.

To make the move procedure flexible, above perturbation operations, rotation, swap, $\mathcal{R}$ moving, and $\mathcal{B}$ moving, are choose by probability ρ_t, ρ_s, ρ_r, and ρ_b, respectively such that ($0 \leq \rho_t, \rho_s, \rho_r, \rho_b \leq 1$) and $\rho_t + \rho_s + \rho_r + \rho_b = 1$.

5.2 Reachability

We will give the diameter of our solution space. The diameter of a solution space is the steps we must take to transform an arbitrary solution to another arbitrary solution. Two packing are said to be the same if and only if their Q-seq are same and the modules in the rooms with the same room label are same.

Theorem 3 *There is a probability of transforming an arbitrary initial packing to any packing via at most $O(n)$ packing perturbation operations mentioned above.*

Proof: Let the initial packing be P_1 and the target packing be P_3, which correspond to the Q-seq Q_1 and Q_3, respectively. To transform a packing to another, we first transform Q_1 to Q_3, then we make each room contain the same module by swap and rotation operations. It is obvious that two Q-seqs are the same if and only if their corresponding parenthesis trees are the same.

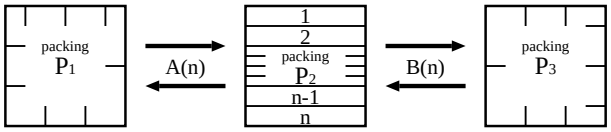
Given an initial Q-seq Q_1 and the target Q-seq Q_3, if there is a special Q-seq Q_2 to which both Q_1 and Q_3 can be transformed via $A(n)$ steps and $B(n)$ steps, then we can deduce that Q_1 can be transformed to Q_3 via at most $A(n) + B(n)$ steps because four packing perturbation operations are reversible. The special Q-seq Q_2 and corresponding packing are shown in Fig.12(a) and Fig.12(b).

Because every Q-seq has the same Abe-order of rooms, we just consider how to make all the positional symbols in two Q-seqs are on the same positions.

There are $n-1$ intervals where n is the number of rooms in the Q-seq. We alter $I(k)$ one by one by the reverse of Abe-order of rooms, namely, we first alter $I(n-1)$, then $I(n-2)$, and so on. We use $I_1(k)$ and $I_2(k)$ to represent $I(k)$ in Q_1 and Q_2, respectively. For convenience to refer, $HR(k)$ is defined as a special operation on a Q-seq. If $k = n$, $\mathcal{R}_k$ is moved to the head of the Q-seq, that is to say, $\mathcal{R}_k$ becomes the first positional symbol in the Q-seq, otherwise, $\mathcal{R}_k$ is move to the position just behind of $\mathcal{R}_{k+1}$. $HR(k)$ can be implemented by a $A(T,k,j)$ operation, where T is $\mathcal{R}$ parenthesis tree. We can transform Q_1 to Q_2 by the following way.

k is set to be $n-1$, apparently, $I_2(k) = \mathcal{B}_{k+1}$. For $I_1(k)$, there are three cases.

case 1: $I_1(k) = \mathcal{B}_{k+1}$. Since $I_1(k) = I_2(k)$, we just need operate $HR(k+1)$ to move $\mathcal{R}_{k+1}$ forward.



(a) transformation process

$$Q_2 = \{\mathcal{R}_n\mathcal{R}_{n-1}\cdots\mathcal{R}_2\mathcal{R}_1\mathcal{B}_1 1\mathcal{B}_2 2\mathcal{B}_1 3\cdots n-1\mathcal{B}_n n\}$$

(b) the Q-Seq of P_2

Figure 12: Transform P_1 to P_3 through P_2

case 2: $I_1(k)$ includes just $\mathcal{R}_{k+1}$. We first move $\mathcal{R}_{k+1}$ by $HR(k+1)$, then $\mathcal{B}_{k+1}$ is moved here to fill the blank which is implemented by an $F(T,k+1,k+1)$ operation, where T is $\mathcal{B}$ parenthesis tree.

case 3: $I_1(k)$ includes more than one $\mathcal{R}$ positional symbols where $\mathcal{R}_{k+1}$ is the last one. One by one, from left to right, $|I_1(k)|$ positional symbols are moved forward by $HR(i)$, where $i \in \{j \mid \mathcal{R}_j \in I_1(k)\}$. Then, $\mathcal{B}_k + 1$ is move here to fill the blank which is implemented by a $F(T,k+1,k+1)$ operation, where T is $\mathcal{B}$ parenthesis tree.

Set $k = k-1$, then iterate until $k = 0$.

Finally, $I_1(k) = I_2(k), k = 1,2,\cdots,n-1$, Q_1 becomes Q_2. In all iterations, each positional symbol at most is moved once. Therefore, it needs at most $2n$ times positional symbol movings to transform Q_1 to Q_2. So, $A(n) = 2n$. Similarly, we have $B(n) = 2n$. Then, we deduce to $A(n)+B(n) \leq 4n$.

After two Q-seqs become the same, we exchange the modules in two rooms in packing P_1 and rotate the module if necessary. Apparently, it needs at most $2n$ steps to make each room in two packings P_1 and P_3 have the same module with the same width and height, so two packing become the same. So, it needs at most $6n$ steps to transform an arbitrary initial packing to another arbitrary packing. □

The theorem says that there is a probability of transforming an arbitrary initial packing to any packing via at most $O(n)$ packing perturbation operations.

5.3 Solution space

An upper bound of the packing solution space of the enhanced Q-seq is $2^{6n}(2n)!/n!$. For accurate formula about the solution space of the Q-seq and analysis, see [12].

6 Experimental results

Utilizing the enhanced Q-seq, we have implemented a packing algorithm controlled by simulated annealing in C++ language on a Sun Ultra60 workstation. Five MCNC building block benchmarks: ami49, ami33, hp, Xerox, and apte are used. We compare our experimental results in chip area with the results of O-tree[4], B*-tree[2], enhanced O-tree[11], and CBL[5].

All initial packings used in the experiments are generated by the same way, where all rooms are piled in one column,

Circuit	Total	O-tree		$B^* - tree$		enhanced O-tree		CBL		enhanced Q-Seq	
	size	area	time	area	time	area	time	area	time	area	time
apte	46.56	47.1	38	**46.92**	7	**46.92**	11	NA	NA	**46.92**	0.35
xerox	19.35	20.1	118	**19.83**	25	20.21	38	20.96	30	19.93	**3.6**
hp	8.30	9.21	57	**8.947**	55	9.16	19	-	-	9.03	**3.5**
ami33	1.16	1.25	1430	1.27	3741	1.24	118	1.20	**36**	**1.194**	40
ami49	35.4	37.6	7428	36.80	4752	37.73	406	38.58	65	**36.75**	57
comparison		1.8%		1.1%		1.8%		-		-	

Table 1: Area(mm^2) and runtime(sec.) comparisons / bold figure: best

that is to say there is only one room located on top boundary and bottom boundary, respectively, and the top n (number of module) rooms contain n modules. The area of a packing is measured by that of the minimum bounding box enclosing the packing.

The area and runtime comparisons with O-tree, B*-tree, enhanced O-tree and CBL are listed in Table 1. This shows that the enhanced Q-seq obtained the average of improvements by 1.8%, 1.1%, and 1.8% in area compared with O-tree, B*-tree and enhanced O-tree, respectively, while using much less time.

7 Conclusion

We have successfully enhanced the Q-seq by empty room insertion and a powerful move procedure based on two parenthesis trees. Compared with other algorithms, the enhanced Q-seq could find better results in much shorter time. It owes to two reasons, one is that it costs at most $O(n)$ time to perturb and decode a Q-seq, and the other is that the diameter of solution space of the enhanced Q-seq is small. Further study on the property of Q-seq and its applications on placement are undergoing.

Acknowledgement

The authors are grateful to Prof. Fujiyoshi and Mr. Kodama, Tokyo University of Agriculture & Technology and Dr. Takashima, Japan Advanced Institute of Science and Technology, for their valuable discussions. This work is part of a project of CAD21 at Tokyo Institute of Technology and is supported in a Research and Development Aid of Support Center for Advanced Telecommunications Technology Research.

References

[1] M. Abe, "Covering the square by squares without overlapping," Journal of Japan Mathematical Physics, Vol. 4, No. 4, pp. 359–366, 1930 (in Japanese).

[2] Y. -C. Chang, Y. W. Chang, G. -M. Wu, and S. W. Wu, "B*-Tree: A New Representation for Non-Slicing Floorplan," Proc. of 37th DAC, pp. 458–463, 2000.

[3] P. S. Dasgupta, S. Sur-Kolay, and B. B. Bhattacharya, "A Unified Approach to Topology Generation and Optimal Sizing of Floorplans, Trans. on CAD, Vol. 17, No.2, pp. 126–1335, 1998.

[4] P. N. Guo, C. K. Cheng, and T. Yoshimura, "An O-Tree Representation of Non-Slicing Floorplan and Its Applications," Proc. 36th DAC, pp. 268–273, 1999.

[5] X. Hong, S. Dong, Y. Ma, Y. Cai, C. -K. Cheng, and J. Gu, "Corner Block List: An efficient topological representation of Non-Slicing floorplan," Proc. of ICCAD 2000, pp. 8–12, 2000.

[6] C. Kodama and K. Fujiyoshi, "Selected Sequence-Pair," Technical Report of IEICE, VLD2001-17, Vol. 101, No. 46, pp. 65–72, 2001 (in Japanese).

[7] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair," Trans. on CAD, Vol. 15, No. 12, pp. 1518–1524, 1996.

[8] H. Murata, K, Fujiyoshi, T. Watanabe, and Y. Kajitani, "A Mapping from Sequence-Pair to Rectangular Dissection," Proc. of ASPDAC '97, pp. 625–633, 1997.

[9] S. Nakatake, H. Murata, K. Fujiyoshi, and Y. Kajitani, "Module Packing Based on the BSG-Structure and IC Layout Applications," Trans. on CAD, Vol. 17, No. 6, pp. 519–530, 1998.

[10] R. H. J. M. Otten, "Automatic floorplan design," Proc. of 19th DAC, pp. 261–267, 1982.

[11] Y. Pang, C. K. Cheng, and T. Yoshimura, "An Enhanced Perturbing Algorithm for Floorplan Design using the O-Tree Representation," Proc. of ISPD 2000, pp. 168–173, 2000.

[12] K. Sakanushi and Y. Kajitani, "The Quarter-State Sequence (Q-Sequence) to Represent the Floorplan and Applications to Layout Optimization," Proc. of IEEE Asia Pacific Conference Circuits And Systems 2000, pp. 829–832, 2000.

[13] T. Takahashi, "A New Encoding Scheme for Rectangle Packing Problem," Proc. of ASPDAC 2000, pp. 175–178, 2000.

[14] M. Tsuboi, C. Kodama, K. Fujiyoshi, K. Sakanushi, and A. Takahashi, "Linear Time Decodable Rectangular Dissection to Represent Arbitrary Packing using Q-Sequence," Proc. of The Tenth Workshop on Synthesis And System Integration of MIxed Technologies 2001. pp. 272–278, 2001.

[15] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design," Proc. of 23rd DAC, pp. 101–107, 1986.