VLSI/PCB Placement with Obstacles Based on Sequence Pair

Hiroshi Murata, *Member, IEEE*, Kunihiro Fujiyoshi, *Member, IEEE*, and Mineo Kaneko, *Member, IEEE*

***Abstract*— In a typical very large scale integration/printed circuit board (VLSI/PCB) design, some modules are preplaced in advance, and the other modules are requested to be placed without overlap with each other and with these preplaced modules. The presence of such obstacles introduces inconsistency to a coding scheme, called sequence pair, which has been proposed for an obstacle free placement problem. We solve this difficulty by proposing a procedure called "adaptation" which transforms an inconsistent sequence pair to a consistent one with the utmost consideration for minimizing the modification. It is shown that a simulated annealing is well organized so that it tests only feasible placements by the adaptation procedure. As a design example, a Microelectronics Center of North Carolina (MCNC) benchmark data "ami49" is packed with treating ten modules among 49 modules as preplaced ones. Further, a PCB example which includes 32 free modules and four preplaced modules (connectors) is laid out successfully by our method with a conventional wiring estimation followed by a commercial router.**

***Index Terms*—Obstacle, placement problem, preplaced module, sequence pair.**

I. Introduction

In very large scale integration (VLSI) design, it often happens that the locations of some macro cells, such as RAM, ROM, and central processing unit (CPU) core, are fixed *a priori* and the other components are subject to be placed in the rest of the chip area. Also in printed circuit board (PCB) design, it is common that the exact coordinates of connectors are determined before designing the placement of the other components. We formulate these situations as a problem called rectangle packing with preplaced rectangles (RPP). Not only the circuit components, but also rectangular obstacles in any type, are candidates to be modeled as preplaced modules. For example, a rectilinear substrate and holes of the substrate can be also modeled with the use of preplaced modules, and components to be placed are requested not to overlap with preplaced modules which correspond to prohibited areas.

Chi [1] studied a similar problem for standard cell layout, in which all the free modules are restricted to have a regular height. Force directed relaxation method (FDR) [2], [3] can be easily tailored to handle the obstacles, but the method has inherent drawbacks in the sensitiveness to the initial placement and in the incompleteness of the overlap elimination. The most practical way would be to use a stochastic algorithm, such as simulated annealing or genetic algorithm, if a proper coding scheme is available.

A coding technique for the slicing structure [4] is not useful for rectangle packing with preplaced rectangles RPP since the preplaced modules might be given nonslicibly. For general placements including both slicible and nonslicible cases, two coding schemes are recently proposed, namely, sequence pair [5], [6] and bounded sliceline grid (BSG) [7], [8]. Using either sequence pair or BSG, it is easy to generate nonoverlapping placements of all the modules by encoding all the (free and preplaced) modules, but the code could be inconsistent to recover the locations of the preplaced modules.

In this paper, we discuss RPP based on the coding scheme of the sequence pair and present a procedure called adaptation which changes a sequence pair so that it becomes consistent to the preplaced modules. The sequence pair is employed here instead of BSG because of the conciseness of the presentation. The adaptation procedure only changes the positions of preplaced modules in a sequence pair and it runs in $O(n^2)$ time where n is the total number of preplaced modules and free modules. Two alternative ways to incorporate the adaptation procedure in a simulated annealing are demonstrated through experiments on an Microelectronics Center of North Carolina (MCNC) building block benchmark example named *ami49*. One of these methods is applied also to a PCB example with a standard wiring length estimation. The resultant placement has turned out to be successfully routed by a commercial router.

The organization of this paper is as follows. Section II gives a formal definition of RPP. Section III addresses that a sequence pair can be inconsistent to RPP. In Section IV, the adaptation procedure is presented. Section V is devoted to the experiments. Section VI is for conclusions.

II. Preliminary

A. Rectangle Packing with Preplaced Rectangles (RPP)

A *module* is a rectangle. A *packing* of a set of modules is a nonoverlapping placement of the modules. The coordinates of a module are the coordinates of the lower left corner of the module. A *preplaced* module is a module whose coordinates as well as width and height are specified. A *free* module is a module whose width and height are specified but the

Manuscript received June 20, 1997. This work was supported in part by Research Body CAD21 at JAIST. This paper was recommended by Associate Editor M. Sarrafzadeh.

H. Murata was with Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan. He is now with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, USA (e-mail: murata@eecs.berkeley.edu).

K. Fujiyoshi is with the Department of Electronic and Information Engineering, Tokyo University of Agriculture and Technology, Japan.

M. Kaneko is with the School of Information Science, Japan Advanced Institute of Science and Technology, Japan.

Publisher Item Identifier S 0278-0070(98)02046-6.

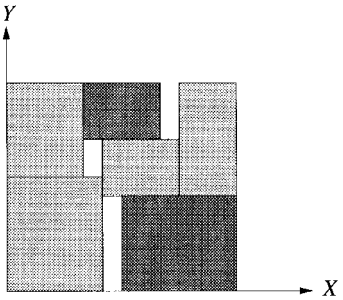


Fig. 1. Feasible packing example. The dark rectangles are the preplaced modules.

coordinates are not specified. Set P of preplaced modules is given such that no two preplaced modules overlap each other and all of them lie in the first quadrant of the plane. Set F of free modules is also given. A *feasible* packing of $P \cup F$ is a packing of $P \cup F$ on the first quadrant of the plane such that all the modules in P are placed at their specified locations. The evaluation of a feasible packing is the area of the minimum bounding rectangle whose lower left corner is at the origin of the plane. The RPP problem treated in this paper is to find the best feasible packing of $P \cup F$. Fig. 1 shows a feasible packing of an instance of RPP.

When all the modules are free, i.e., $P = \emptyset$, then RPP coincides with a known packing problem, denoted by RP, which is proved to be $\mathcal{NP}$ hard [5], [6]. Thus, RPP is also in $\mathcal{NP}$-hard class.

B. Sequence Pair

For the free packing problem (RP), a useful coding scheme is proposed in [5] and [6] as follows.

A *sequence pair* for a set of n modules is a pair of sequences of the n module names. For example, (abc, bac) is a sequence pair for module set $\{a, b, c\}$. It is easily understood that the variety of the sequence pair for n modules is $(n!)^2$.

A sequence pair imposes a horizontal/vertical (H/V) constraint for every pair of modules as follows:

$(\cdots a \cdots b \cdots, \cdots a \cdots b \cdots) \rightarrow a$ should be placed to the left of b

$(\cdots b \cdots a \cdots, \cdots a \cdots b \cdots) \rightarrow a$ should be placed below b.

For example, sequence pair (abc, bac) imposes a set of H/V constraints: $\{a$ should be placed to the left of c, b should be placed to the left of c, b should be placed below $a\}$.

The H/V constraints of a sequence pair can be intuitively grasped using the *oblique-grid* notation. For example, Fig. 2(a) shows the oblique grid of sequence pair (abc, bac). It is an $n \times n$ grid obliquely drawn on the plane which is constructed so that the first sequence is observed when one reads the module names on the positive slope lines from left to right and the second sequence is observed similarly with respect to the negative slope lines. It shows the H/V constraints: module c is in the right quarter view range (between $-45°$ and $+45°$) of module a on the oblique grid, then c should be placed to the right of a.

It has been proven in [5] and [6] that the set of H/V constraints imposed by every sequence pair is satisfiable, and

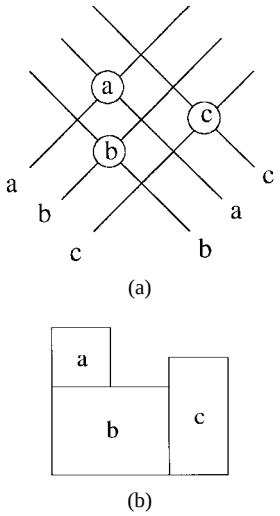


Fig. 2. Oblique grid notation of a sequence pair (abc, bac) and the free realization of the sequence pair. (a) Oblique grid notation sand and (b) free realization.

an area minimum packing under the constraint can be obtained in polynomial time, and further, there is a sequence pair which leads an (globally) area minimum packing. Then, the sequence pair is easily utilized as a coding scheme of a stochastic algorithm.

To construct an area minimum placement for a sequence pair, one-dimensional compaction is carried out under the H/V constraints of the sequence pair. The modules are greedily pushed to the left and to the bottom as shown in Fig. 2(b). The resultant placement is called the *free realization* of the sequence pair.

The free realization can be obtained in $O(n^2)$ time[1] by using the H/V constraint graph which is constructed faithfully to the H/V constraints. More in detail, Step 1 constructs a vertex weighted directed acyclic graph whose vertex set corresponds to the modules and whose edge set corresponds to the horizontal constraints in the direction from left to right. The weight of each vertex is the width of the corresponding module. Determine the X coordinate of each module by the longest path length from the source nodes to the node of the module. Step 2 determines the Y coordinate of each module in a similar way using the vertical constraints in the direction from bottom to top.

III. Feasibility of Sequence Pair for RPP

Now we assume $P \neq \emptyset$, i.e., one or more modules are preplaced. If we only encode the free modules into a sequence pair, a free module and a preplaced module can easily overlap each other in the free realization of the sequence pair since the preplaced module is totally ignored. Fig. 3(a) illustrates an example for this situation. In the figure, free modules a, b, c, d are placed without considering the preplaced modules x and y. As a result, d and b overlap with y.

We employ an alternative approach, that is, to encode both the free modules and the preplaced modules into a sequence pair. One difficulty in the free realization of such

[1] The time complexity is reduced to $O(n \log n)$ by [9].

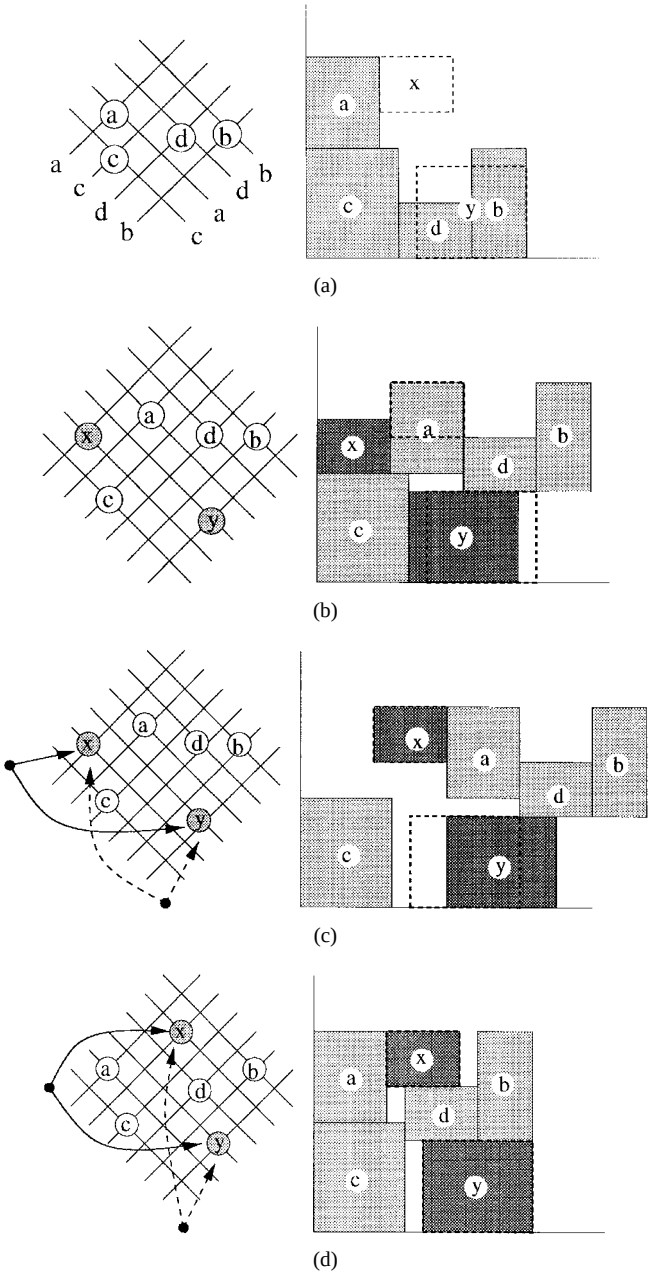


Fig. 3. Feasibility of sequence pair. (a) Sequence pair of free modules, (b) sequence pair of free and preplaced modules, (c) additional constraints for preplaced modules, and (d) feasible sequence pair.

a sequence pair is that the X (as well as Y) coordinate of a preplaced module may be set too small because modules are compacted to the left (and to the bottom) without considering the preassigned coordinates. Fig. 3(b) shows such an example where the X and Y coordinates of preplaced module x are set too small and the X coordinate of preplaced module y is set too small. This difficulty is easily solved by introducing an additional constraint for each preplaced module: "The X (Y) coordinate of each preplaced module should not be less than the specified value."

The free realization procedure is easily modified to handle the additional constraints without increasing the asymptotic complexity by adding additional edges from source to the preplaced modules with the preassigned coordinates as their weights in the H/V constraint graph [Fig. 3(c)]. The resultant placement is now called the *propped realization*, named from an intuitive image of the additional constraints.

Fig. 3(c) shows the propped realization for the same sequence pair used in Fig. 3(b). The arcs with solid lines in the oblique grid denote the additional constraints in the horizontal constraint graph and the arcs with broken lines denote those in the vertical constraint graph. In the figure, preplaced module x is placed at the specified location. However, preplaced module y is not placed at the specified location (the X coordinate is set too large). This is because the additional constraints introduced above cannot prevent the coordinates from being set too large. It is impossible to reform the X coordinate of module y by adding other constraints since the sequence pair primarily imposes the constraint: "y should be placed to the right of x." In the following, a sequence pair is said to be *feasible* when its propped realization is feasible, otherwise it is said to be *infeasible*.

It is concluded that a sequence pair is not always feasible for RPP. However, it is still useful from the following fact.

Lemma 1: For any instance of RPP, there is a feasible sequence pair. Furthermore, there is a sequence pair which leads to an optimal solution of the problem.

Proof: A proof for the second claim suffices as a proof for the first claim. It is clear that there is an optimal solution Π for RPP with preplaced modules P and free modules F. From [5], [6], there is a sequence pair S for $P \cup F$ such that the length of every path between two vertices in the H (V) constraint graph is not greater than the X (Y) distance between the modules in Π if the path consists of H (V) constraint edges only, where the length of the path refers to the sum of the weights of all the edges and vertices in the path except for the two vertices at the ends. Since the weight of each additional edge equals to the X (Y) coordinate of the preplaced module, the length of every path between two vertices in the H (V) constraint graph is not greater than the difference between X (Y) coordinates of the modules in Π. Therefore, in the propped realization of S we obtain a placement which is not worse than Π thus also optimal. □

Fig. 3(d) shows an example of a feasible (and optimal) sequence pair and its propped realization.

IV. Adaptation

Among $(n!)^2$ sequence pairs there are feasible sequence pairs including one for the optimal solution of RPP and infeasible sequence pairs. We should consider how to treat the infeasible sequence pairs when exploring the solution space by a stochastic algorithm. The simplest way would be to evaluate them infinitely negative, but the smoothness of the search would be strongly weakened. To keep the smoothness as much as we can, it is desirable that each infeasible sequence pair is evaluated equally to a feasible sequence pair which resembles the infeasible one. For this purpose, we present a procedure called *adaptation* which transforms a given sequence pair to a feasible sequence pair by changing only the positions of preplaced modules.

A. Necessary Condition

Suppose unit square modules a and b are preplaced at (3, 3) and at (5, 5), respectively. According to the gridding strategy, we can interpret this situation with a little ambiguity as "a is left of b" or "a is below b." Whichever one of these two interpretations we employ, "a appears before b" in the second sequence of the corresponding sequence pair. More in general, we describe a necessary condition for the second sequence of a feasible sequence pair.

For any two preplaced modules a and b, a is said to *dominate* b if

$$x(a) < x(b) + w(b) \quad \text{and} \quad y(a) < y(b) + h(b)$$

where $x(a)$ and $y(a)$ are X and Y coordinates of module a and $w(a)$ and $h(a)$ are width and height of module a, respectively.

The domination relation introduces a partial order into the module set. Then, the second sequence of a sequence pair is now said to be *topologically sorted* if it satisfies the condition: For any two preplaced modules a and b, if a dominates b, then a appears prior to b in the second sequence. It is clear that the following lemma holds.

Lemma 2: It is necessary for a sequence pair being feasible that the second sequence is topologically sorted.

B. Algorithm

In the propped realization of a sequence pair, the X and Y coordinates of a module are determined only by the preceding modules in the second sequence. It turns out that the coordinates of a module can be determined one by one, traversing the second sequence. Then, we design our adaptation procedure so that it iteratively "tests and virtually places" a module according to the second sequence. Thus, after some iterations, some of free modules would be virtually placed. In the following, we use the relation "domination" for those virtually placed free modules as well as for the preplaced modules. See the algorithm at the bottom of the page.

C. Illustrative Example

A behavioral example is presented for the instance of RPP shown in Fig. 4(a), in which two dark modules x and y are preplaced modules and the other a, b, c, d are free modules.

Suppose sequence pair $(acdbyx, cadxyb)$ is given as the input sequence pair. Fig. 4(b) shows this initial sequence pair. In Step 1), x and y are exchanged in the second sequence and the sequence pair becomes $(acdbyx, cadyxb)$ as shown in Fig. 4(c).

Procedure `Adaptation`
Input: A set of preplaced modules, a set of free modules, a sequence pair.
Output: A feasible sequence pair.
Step 1) Topologically sort the second sequence such
that no change occurs when it is already topologically sorted.
Step 2) `For` $k = 1, 2, \cdots, n$, `do`
Step 2.1) `if` (the kth module a in the second sequence is a free module) `then`
Step 2.1.1) Temporary set the coordinates of a according to the propped-realization of a.
Step 2.1.2) `if` (there exists a preplaced module which dominates a, in the last $n - k$ modules of the second sequence) `then`
Determine a preplaced module q such that q directly or transitively dominates a, and there is no preplaced module which dominates q in the last $n - k$ modules of the second sequence. In the second sequence, move q just before a. After that, a is used for referring q. (since q is now the kth module in the second sequence.)
`endif`
`endif`
Step 2.2) `if` (the kth module a in the second sequence is a preplaced module) `then`
Let (x, y) be the coordinates of a in the propped-realization. Let $(x(a), y(a))$ be the specified coordinates of preplaced module a.
`If` $(x > x(a))$ `then`
move a minimally toward the top in the first sequence so that $x \leq x(a)$ holds.
`else if` $(y > y(a))$ `then`
Move a minimally toward the end in the first sequence so that $y \leq y(a)$ holds.
`endif`
`endif`
`endfor`

(**Procedure** `Adaptation` **End**)

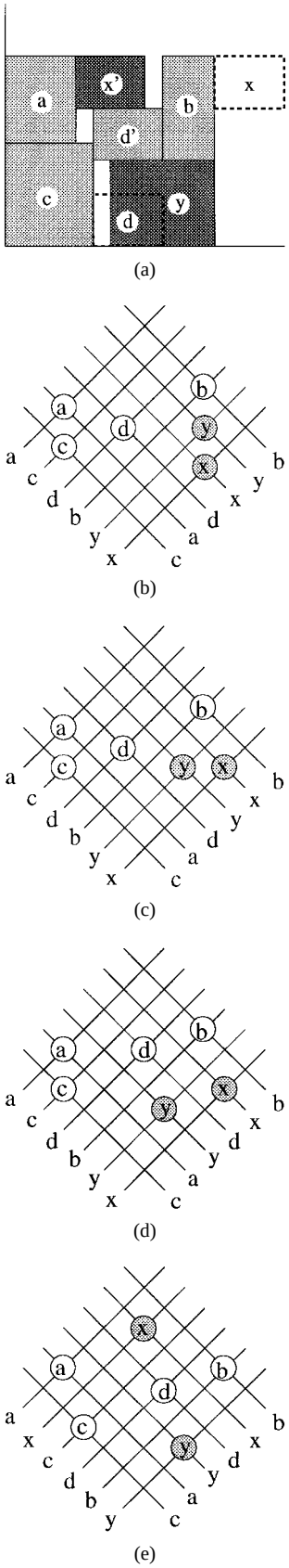


Fig. 4. Snapshots of the adaptation procedure. Part (a) shows the packing corresponding to the sequence pair shown in (e) and (b) shows the input sequence pair. In (c), y is brought before x in the second sequence. In (d), y is brought before d in the second sequence. In (e), x is brought before c in the first sequence. (a) The packing example, (b) $(acdbyx, cadxyb)$, (c) $(acdbyx, cadyxb)$, (d) $(acdbyx, caydxb)$, and (e) $(axcdby, caydxb)$.

Now, Step 2) begins. In iteration $k = 1$ and in $k = 2$, free modules c and a are processed because they are the first and the second modules in the second sequence, respectively. They are virtually placed at the positions shown in Fig. 4(a).

In iteration $k = 3$, module d is once tried to be placed at the right of c in Step 2.1.1), as indicated by the rectangle with dotted lines which is marked "d" in Fig. 4(a). Then, module y is selected and brought before d in the second sequence in Step 2.1.2). As a result, the sequence pair becomes $(acdbyx, caydxb)$ as shown in Fig. 4(d). Module y is virtually placed at its specified position, as it is marked "y" in Fig. 4(a).

In iteration $k = 4$, module d is placed at the position marked "d'" in Fig. 4(a).

In iteration $k = 5$, preplaced module x is processed. In Step 2.2), it is once tried to placed at the position indicated by the rectangle with dotted lines which is marked "x" in Fig. 4(a) since it is the position imposed by the sequence pair shown in Fig. 4(d). Then the X coordinate is found too large and module x is moved toward the top of the first sequence and put between a and d. The sequence pair becomes $(axcdby, caydxb)$ which is illustrated in Fig. 4(e). Module x is virtually placed at its specified position, as it is marked "x'" in Fig. 4(a).

This sequence pair is not changed in iteration $k = 6$ and becomes the output of the procedure.

One can examine on this example that the resultant sequence pair is feasible. It is important to note that the H/V constraints among the free modules are preserved.

D. Proof of Adaptation

Theorem 1: The adaptation procedure changes the given sequence pair such that:

1) the output sequence pair coincides with the input sequence pair if the given sequence pair is feasible;
2) the output sequence pair is always feasible;
3) the H/V constraints with respect to the free modules are preserved;
4) the procedure runs in $O(n^2)$ time, where n is the total number of the preplaced modules and the free modules.

Proof: Items 1) and 3) are easily understood. Items 2) and 4) are proved in the following.

2): `Adaptation` outputs a feasible sequence pair.

Let M be the set of all the modules. Let M^k denote the set of first k modules in the second sequence of S. Let S^k denote the sequence pair for M^k obtained from a sequence pair S for M by deleting all the modules in $M - M^k$.

Assume that when the procedure reaches the beginning of the kth iteration in Step 2), the second sequence is topologically sorted with respect to the domination relation between preplaced modules, and also assume that S^{k-1} is feasible (for M^{k-1}). Both assumptions are satisfied trivially for $k = 1$. It is clear that Step 2.1) through Step 2.2) determines the kth module in the second sequence and its position in the first sequence without changing S^{k-1}. Then, it suffices as a proof if we show Step 2.1) through Step 2.2) produces a feasible S^k (for M^k) in kth iteration.

The proof is by contradiction. Assume that the kth module a in the second sequence and its position in the first sequence are determined in the kth iteration and S^k becomes infeasible.

Module a should be a preplaced module since S^k can not be made infeasible if a is a free module.

It is also true that S^k can not be made infeasible if Step 2.2) is successfully executed. So, there must be a module b in M^{k-1} such that

$$x(a) < x(b) + w(b) \quad \text{and} \quad y(a) < y(b) + h(b).$$

If b is a preplaced module, since b had been after a in the second sequence at the end of Step 1), it was brought before a afterward in Step 2.1.2) in the iteration k' where $k' < k$. In the iteration k', the position of b in the second sequence was changed because b is determined as module q in Step 2.1.2), hence b is not dominated by any other module. This contradicts the fact that a dominates b. Then b should be a free module determined in the iteration k''. In the iteration k'', b is tested to be not dominated by any preplaced module in Step 2.1.2). This contradicts the fact that a dominates b. Hence, it is concluded that S^k is feasible. Therefore, the resultant sequence pair obtained by the procedure is feasible.

4): The adaptation procedure can be run in $O(n^2)$ time.

Step 1) can be done in $O(n^2)$ time as follows.

a) Construct an $n \times n$ matrix D such that

$$D(a,b) = \begin{cases} 1, & \text{if module } a \text{ directly or transitively} \\ & \text{dominates } b \\ 0, & \text{otherwise.} \end{cases}$$

D can be constructed in $O(n^2)$ time by sorting the modules by their coordinates in X and in Y. This can be understood by examining the following fact: If a transitively dominates b and $x(a) < x(b)$ and $y(a) \geq y(b) + h(b)$, then there is a series of modules from a to b dominating one after another and their X coordinates are monotonically increasing.

b) Topologically sort the second sequence using D. This can be done also in $O(n^2)$ time by a selection sort algorithm.

(The claim is easily proven also by the ordinary depth first algorithm, but we use the above algorithm with utmost consideration for minimizing the modification.)

Step 2.1) can be done in $O(n)$ time by traversing the modules reversely in the second sequence. Step 2.2) can be done in $O(n)$ time by introducing two arrays $X[1 \cdots n]$ and $Y[1 \cdots n]$ which holds a "negative locus" [5], [6] of the kth module in the second sequence after the module is virtually placed. More in detail, each element in the array is maintained to have the minimum value to satisfy the next condition: $X[1 \cdots n]$ holds a nondecreasing series, $Y[1 \cdots n]$ holds a nonincreasing series, and finally, for each module a in the first k modules in the second sequence, $X[j] \geq x(a) + w(a)$ and $Y[j] \geq y(a) + h(a)$ where j is the a's position in the first sequence.

Step 2.1) through Step 2.2) is repeated n times in Step 2). Therefore, the adaptation can be done in $O(n^2)$ time.

V. Experiments

A. Simulated Annealing with Adaptation

The adaptation procedure is implemented in a standard simulated annealing to solve RPP. The outline of the simulated annealing is as follows.

Procedure SA

Input: Set P of preplaced modules, set F of free modules, number of iterations, initial temperature, and final temperature.

Output: A feasible packing of $P \cup F$.

Step 1) Generate a random sequence pair, initialize the loop counter, set temperature to the initial value, and set the decreasing ratio of the temperature such that it reaches to the final temperature when the loop counter reaches to the limit.

Step 2) If the loop exceeds the given limit, output the best packing obtained so far and then stop.

Step 3) Apply one of the following three move operations to alter the sequence pair.

- Exchange two module names in the first sequence.
- Exchange two module names both in the first sequence and the second sequence.
- Exchange the width and the height of a module.

Step 4) Evaluate the sequence pair by the area of the corresponding placement.

Step 5) If the evaluation is improved or not changed, then accept the change. Otherwise, accept the change stochastically depending on the temperature and on the difference of the evaluations.

Step 6) Decrease the temperature exponentially by the ratio obtained in Step 1) and go to Step 2).

(**Procedure SA End**)

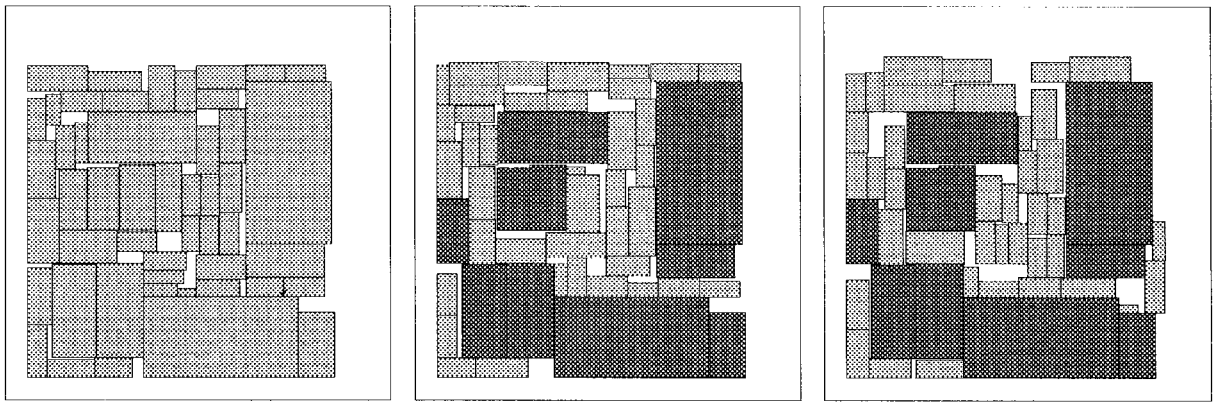
The adaptation can be used in one of the following two manners.

- Adapt-in-Eval: Use the adaptation procedure internally in the evaluation step Step 4), intending to evaluate an infeasible sequence pair as an equivalent of a feasible sequence pair which is generated from the original sequence pair by the adaptation.
- Adapt-in-Move: Use the adaptation procedure internally in the initialization step Step 1) and in the move step Step 3) so that an infeasible sequence pair is never generated.

The major difference of the two methods would be as follows. In both methods, the SA explores the space of $(n!)^2$ sequence pairs. Suppose there is a cluster of infeasible sequence pairs in the solution space. The search may go into the cluster in the Adapt-in-Eval method, while in the Adapt-in-Move method it is flicked out from the cluster right after an infeasible sequence pair is generated eventually. Then, the reachability to an optimal solution is guaranteed in the Adapt-in-Eval method and not in the Adapt-in-Move method.

B. Packing Experiments

An MCNC example called *ami49* is used as the input data. As a preliminary run, all the 49 modules are treated as free modules and they are packed by the simulated annealing. After



(a) (b) (c)

Fig. 5. Three packings of ami49. Part (a) is obtained by RP. Parts (b) and (c) are obtained by Adapt-in-Eval method and Adapt-in-Move method, respectively, setting the dark modules being preplaced.

TABLE I
AN EXPERIMENTAL RESULT OF ami49 (AVERAGE OF 10 RUNS)

	RP	Adapt-in-Eval	Adapt-in-Move
Area (mm^2)	37.978808	38.353762	40.062617
Time (sec)	331.78	1004.15	949.68

TABLE II
AN EXPERIMENTAL RESULT OF DEMOSMD

	MST	**MBOX**
Total wiring length (mm)	7,914.64	8,188.96
Number of vias	368	382
Placement time (sec.)	630	150
Routing time (sec.)	568	590
Total time (sec.)	1,198	740

that, the first ten biggest modules, with respect to the areas, are specified as preplaced modules whose coordinates are fixed to those obtained by the preliminary run to yield a RPP version of *ami49*. The reason why these modules are selected is because of our experience that the obstacles are usually not many, but big. The Adapt-in-Eval method and Adapt-in-Move method are applied to the same input, the RPP version of *ami49*.

Fig. 5 shows the resultant packing of these methods and the one of preliminary run. Table I shows the average performance of ten runs of the Adapt-in-Eval method, Adapt-in-Move method, and also the preliminary run in the column RP for reference. Every trial is performed on a Sun SS-5 (75 MHz).

The run time in the preliminary run was the shortest, as in Table I. This would be natural because the adaptation is not executed and also because Step 4) runs in $O(n \log n)$ time when there is no preplaced modules with an algorithm similar to [9]. The reason why Adapt-in-Move runs faster than Adapt-in-Eval method would be because the input sequence pair of the adaptation in Adapt-in-Move method is feasible or almost feasible, while that of Adapt-in-Eval method can be far from feasible.

It is interesting that RP achieved the minimum area, although the number of the tested sequence pairs is the same as those for Adapt-in-Move and Adapt-in-Eval. It is also observed from Table I that Adapt-in-Eval method achieves better than Adapt-in-Move method. This result seems to reflect the reachability difference which is mentioned earlier in this section. From this reachability difference together with the above experimental result, we can conclude that Adapt-in-Eval method is preferable for practical applications.

C. Place and Route Experiments

Based on Adapt-in-Eval method, a PCB example is placed with setting the connectors preplaced.

Wires are considered additionally to the area in the evaluating function. To keep the wiring space, the widths and the heights of the modules are uniformly enlarged. After the packing is obtained from a sequence pair, the minimum spanning tree is calculated for each net, and the sum of the length of each edge in the tree is used as an estimation of the wiring length of the net. The evaluation of a sequence pair is the area of the packing plus the total sum of the estimated length of every net. Weighting coefficient is introduced so that the terms of the area and the estimated wire length are approximately balanced.

A PCB example which includes 36 modules and 88 multiterminal nets was used in the experiment. The data is called DEMOSMD, which is provided with a commercial layout editor (Protel Advanced PCB 2.8). The simulated annealing was scheduled on this data to try 100 000 moves. The resultant placement was sent to a commercial router (Protel Advanced Route3) to complete the routing using four layers. All design experiments are achieved on a personal computer (Sharp Mebius PC-A355, Pentium 100 MHz).

As a result, all the nets were successfully routed, as shown in Fig. 6. The performance is summarized in the column "MST" in Table II.

The same trial with the wire length estimation by the half perimeter of the minimum bounding box, which is a popular way to save the computational cost for the estimation, was carried out, and it has been shown that the routing is successfully finished on the resultant placement. Results are shown in the column "MBOX" in Table II.

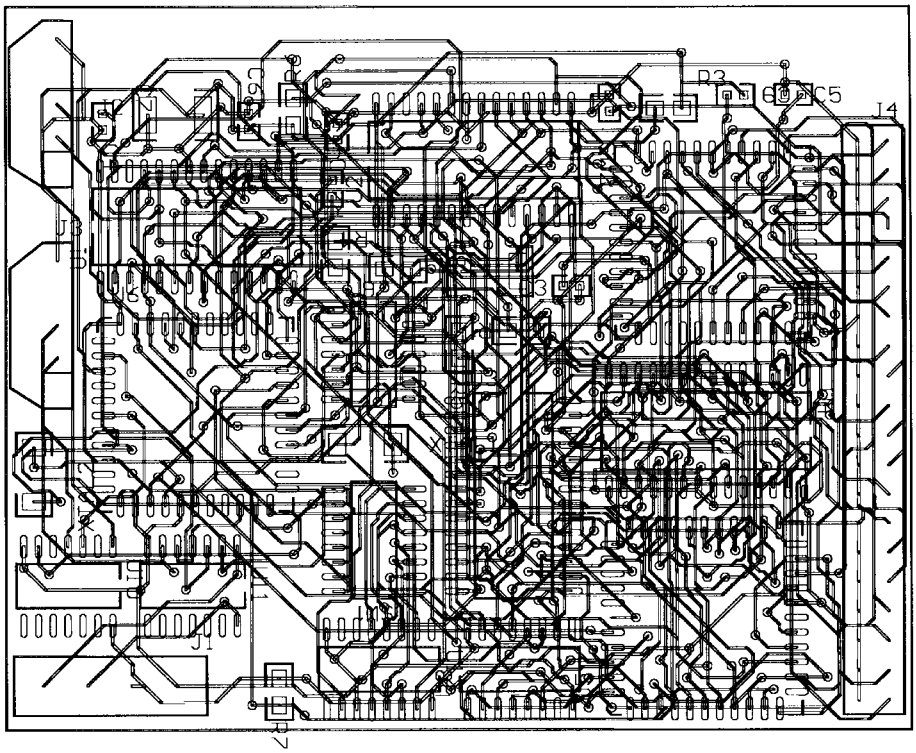


Fig. 6. A place-and-route result of DEMOSMD. Connectors J1, J2, J3, and J4 are preplaced.

VI. Conclusion

We showed that the presence of the preplaced modules introduces an inconsistency to the sequence pair coding scheme. The adaptation procedure is proposed to transform an inconsistent sequence pair to a consistent one. It runs in $O(n^2)$ time, where n is the total number of preplaced modules and free modules, with the following preferable features: 1) the output sequence pair is guaranteed to be feasible in the sense that every preplaced module can be placed at its specified position, 2) a sequence pair is not changed when it is originally feasible, and 3) it does not alter the H/V constraints with respect to the free modules.

Two possible ways to incorporate the adaptation procedure into a simulated annealing are shown, namely Adapt-in-Eval and Adapt-in-Move. A discussion is given which results in the Adapt-in-Eval would be preferable because of the reachability difference and our experimental results supported this argument.

It would be important to study further on, for example, the performance difference between Adapt-in-Eval and Adapt-in-Move for various problem instances and the extending of the adaptation procedure to cope with soft modules.

Acknowledgment

The authors would like to thank Prof. Y. Kajitani and Research Associate S. Nakatake of the Tokyo Institute of Technology for their helpful discussions.

References

[1] M. C. Chi, "An automatic rectilinear partitioning procedure for standard cells," in *Proc. 24th ACM/IEEE Design Autom. Conf.*, pp. 50–55, 1987.

[2] L. Sha and R. W. Dutton, "An analytical algorithm for placement of arbitrarily sized rectangular blocks," in *Proc. 22th ACM/IEEE Design Autom. Conf.*, 1985, pp. 602–608.

[3] A. Alon and U. Ascher, "Model and solution strategy for placement of rectangular blocks in the Euclidean plane," *IEEE Trans. Computer-Aided Design*, vol. 7, no. 3, pp. 378–386, Mar. 1988.

[4] D. F. Wong and C. L. Liu, "A new algorithm for floorplan designs," in *Proc. 23rd ACM/IEEE Design Autom. Conf.*, 1986, pp. 101–107.

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

[6] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence pair," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1518–1524, Dec. 1996.

[7] S. Nakatake, H. Murata, K. Fujiyoshi, and Y. Kajitani, "Bounded-slicing structure for module placement," Tech. Rep. Institute of Electronics, Information, and Communication Engineers (IEICE), vol. VLD94, no. 313, pp. 19–24, 1994.

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

[9] T. Takahashi, "An algorithm for finding a maximum-weight decreasing sequence in a permutation, motivated by rectangle packing problem," Tech. Rep. IEICE, vol. VLD96, no. 201, pp. 31–35, 1996.

Hiroshi Murata (M'95) was born in 1957. He received the Bachelor's degree in electrical engineering from Kanazawa University, Ishikawa, Japan, in 1980. He received the Master's and Doctorate degrees in information science from Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan, in 1994 and 1997, respectively.

He was engaged in business software development from 1980 to 1984 at Computer Applications Co., Ltd., Tokyo, Japan, and in CAD software development for Hybrid IC and PCB circuits from 1984 to 1997 at Komatsu Murata Mfg. Co., Ltd., Ishikawa, Japan. He has been at the University of California, Berkeley, as a Visiting Researcher since April 1997. His research interests are in physical design algorithms, especially in floorplanning and placement.

Kunihiro Fujiyoshi (M'96) was born in Tokyo, Japan, in 1964. He received the B.E., M.E., and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1987, 1989, and 1994, respectively.

From 1992 to 1996, he was a Research Associate at the School of Information Science at Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan. Since 1997, he has been a Lecturer in the Department of Electronic and Information Engineering, Tokyo University of Agriculture & Technology. His research interests are in combinatorial algorithms and VLSI layout design.

Mineo Kaneko (M'90) received the B.E., M.E., and Dr.E. degrees in electrical and electronic engineering, from Tokyo Institute of Technology, Tokyo, Japan, in 1981, 1983, and 1986, respectively.

From 1986 to 1996, he was with the Department of Electrical and Electronic Engineering, Tokyo Institute of Technology, as a Research Associate, a Lecturer, or an Associate Professor. Currently, he is an Associate Professor in the School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan. His research interests include circuit theory and CAD for VLSI's, fault tolerance, and analog and digital signal processing.

Dr. Kaneko is a member of the Institute of Electronics, Information, and Communication Engineers (IEICE), Japan.