Rectangle Packing and Its Applications to VLSI Module Placement Problem

By Hiroshi Murata

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Doctor of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Eiji Okamoto

January 16, 1997

Abstract

The first and the most critical stage in VLSI layout design is the placement. Its background is the *rectangle packing problem* : Given a set of rectangular modules of arbitrary sizes, place them without overlap on a plane within a rectangle of the minimum area. Since the variety of the packing is uncountably infinite, the key issue for successful optimization is the introduction of a finite solution space which includes an optimal solution and excludes all the infeasible solutions. The main contribution of this thesis is in the introduction of such a solution space where each packing is represented by a pair of module name sequences, called *sequence-pair*. The introduction of this solution space enables us to use stochastic optimization method such as simulated annealing, and it is demonstrated that hundreds of modules was packed very efficiently. The biggest MCNC benchmark example is also shown to be placed very promisingly with a conventional wiring consideration method.

Although module positions are successfully represented by the sequence-pair, it is desired frequently in VLSI design that channels are represented together with modules, because a channel router is often used in the following routing stage. For this request, this thesis gives a mapping from a sequence-pair to a rectangular dissection, which represents channels by line segments.

Placement with obstacles in the chip is also discussed in this thesis, for dealing with pre-placed modules and with a rectilinear placement region. The obstacles are easily included in a sequence-pair to eliminate the overlaps, but the sequence-pair cannot guarantee to recover the assigned coordinates of the obstacles. To solve this practical problem, this thesis gives an algorithm which changes an inconsistent sequence-pair to a consistent one.

Acknowledgments

The author indebted to his principal advisor Professor Eiji Okamoto of Japan Advanced Institute of Science and Technology for his constant encouragement.

The author would like to thank to his advisor Associate Professor Kunihiko Hiraishi of Japan Advanced Institute of Science and Technology for his helpful discussions and suggestions.

Special thanks are to Associate Professor Mineo Kaneko of Japan Advanced Institute of Science and Technology for his constant guidance. Much of the final year of research was guided by him.

The author wishes to express his sincere gratitude to his prior principal advisor Professor Yoji Kajitani of Tokyo Institute of Technology for his helpful discussions, guidance, and all the activities, that inspired the author throughout the course of this research.

The author devotes his sincere thanks and appreciation to Research Associate Kunihiro Fujiyoshi of Japan Advanced Institute of Science and Technology, currently at Tokyo University of Agriculture and Technology. His eager discussions helped the author especially on building the proofs presented in this thesis.

Special thanks are to Research Associate Shigetoshi Nakatake of Tokyo Institute of Technology. He jointly started this research with the author, and has been an excellent partner throughout the research. He gave the first idea for our common target, and the discussions on his initial idea helped the author to develop the theory presented in this thesis.

The author is grateful to all who have affected or suggested his areas of research, especially to Visiting Professor Milan Vlach and Visiting Associate Professor Magnus Halldórsson.

Special thanks are to Mr. Eda, Mr. Funahara, Mr. Kaneko, Mr. Yamagishi and all the engineers at Murata Mfg., Co. Ltd. The motivation of this research is mainly from the author's experience in developing the CAD/FA system with them.

Finally, I would like to thank my wife Akiko and our sons Shun'ichi and Tatsurou for their love and support.

Contents

Chapter 1

Introduction

1.1 Rectangle Packing Problem

Layout in physical design of VLSI is, simply to say, to pack all the circuit elements in a chip without violating the design rules, so that the circuit performs well and the production yield is high. Among the variety of targets in different stages, the problem defined as follows is the base of all of them.

> **Rectangle Packing Problem: RP**
> Let $\mathcal{M}$ be a set of n rectangular modules whose heights and widths are given in real numbers. (Orientations are fixed.) A packing of $\mathcal{M}$ is a non-overlapping placement of all the modules. The minimum bounding rectangle of a packing is called the chip. Find a packing of $\mathcal{M}$ in a chip of the minimum area.

Notice that **RP** is not simply a combinatorial optimization problem since the heights and widths of modules are arbitrary real numbers. It will be shown in this thesis that the problem belongs to the $\mathcal{NP}$-hard class.

1.2 Previous Researches

Similar problems have been studied from mathematical interests [1, 2, 3, 4, 5], but they are far from real applications in VLSI layout design. In VLSI design, deterministic algorithms have been used based on heuristic ideas [6, 7], but they easily fall in a non-global local-optimum. An alternative approach is to use stochastic searches, such as simulated annealing and genetic algorithm. A stochastic search is known to have a potential for finding one of the best solutions in the "solution space" in a controlled time [8, 9].

To apply a stochastic search, it is required to reduce the problem into a combinatorial level by introducing a discrete *solution space*. A solution space is a set of codes, each of which represents a construction of placement. A stochastic search is to explore the solution space randomly and heuristically to find a good solution, and to output the best found solution by the end of the given limit of time. However, if the solution space does not include any optimal solution, the found solution cannot be optimal. Unfortunately, an optimal solution is not guaranteed to be included in the most of known solution spaces [10, 11, 12]. The solution space proposed by Onodera, Taniguchi and Tamaru [13] includes an optimal solution but also includes infeasible solutions, thus it is not useful

for a stochastic search. (They uses an exhaustive algorithm, but the size of the tractable problem is limited up to six modules.)

The discussion above concludes that the key issue is to invent a solution space which includes an optimal solution but excludes all the infeasible solutions. Such a solution space is said to be *P-admissible* in this research, though there is no known example for the rectangle packing problem.

1.3 Thesis Outline

The main contribution of this thesis is finding a P-admissible solution space for the rectangle packing problem. The solution space is defined as follows. A pair of module name sequences (for example, $(abcd, bdac)$ for modules a, b, c and d) is called a *sequence-pair*. It is defined that sequence-pair implies a horizontal/vertical (right of, left of, below, above) relation for every pair of modules m and m', depending on whether m is {before, after} m' in the {first, second} sequence. A compaction procedure is given so that it makes a sequence-pair corresponds to a best packing of all the packings that satisfy the implied horizontal/vertical relations. The set of all the sequence-pairs, thus of the cardinality $(n!)^2$, is our solution space. The P-admissibility of this solution space is proved in detail in this thesis. In experiments using the data abstracted from industrial examples, hundreds of modules are effectively packed. Furthermore, to see how the method is promising, an example of tens of modules is shown to be placed with a conventional wiring consideration method.

A sequence-pair represents a non-overlapping placement of modules, but by nature it does not represent any wiring channel. This could be a difficulty in applying the method to IC layout in which the placement is followed by a channel router. In previous researches, a rectangular dissection has been used to represent relative positions of channels as well as relative positions of modules. As a complementary contribution to the sequence-pair method, we give a mapping from a sequence-pair to a rectangular dissection, introducing the minimum number of channels to preserve the information on module positions. The tight upper bound of the number of introduced channels is given with detailed proofs.

As a practical contribution of this thesis, further consideration is devoted to cope with a specific requirement in PCB/VLSI design. In typical PCBs, there are obstacles such as holes and connectors. The obstacles are often found also in VLSI design, for example, pre-placed macro cells. To eliminate the modules being overlapped with such obstacles, these obstacles can be modeled as "pre-placed modules" and included in a sequence-pair. However, it is not guaranteed that such a sequence-pair can recover the assigned coordinates of the pre-placed modules. A procedure is presented to change an arbitrary sequence-pair to fit such environments.

This thesis is organized as follows. In Chapter 2, the sequence-pair is introduced and its P-admissibility is proved. In Chapter 3, a mapping from a sequence-pair to a rectangular-dissection is provided to generate channel information. Chapter 4 is devoted to present an adaptation procedure to tailor the solution space for the obstacles. Finally, Chapter 5 concludes this research with remarks.

1.4 Remarks

The Sequence-Pair idea was born in a research discussion in Kajitani-lab, Japan Advanced Institute of Science and Technology, in 1994, inspired by another idea, called Bounded-Sliceline-Grid (BSG), which is a uniform grid structure but specially designed for VLSI module placement problems. Since then, they have been working together to develop the BSG method and the Sequence-Pair method in parallel, as is listed in the publications section. However, this thesis rarely describes about BSG method, since theory of the sequence-pair is discussed in a complete and closed form. The birth and growth of the BSG and the Sequence-Pair in the very early stage are described in [15] which was written by the then Advisor Professor Y. Kajitani.

Chapter 2

Rectangle Packing Solution Space Composed of Sequence-Pairs

2.1 Introduction

Layout in physical design of VLSI is, in short, to pack all the circuit elements in a chip without violating design rules, so that the circuit performs well and the production yield is high. There are so much variety of targets in different stages but the following problem is the core of them.

Rectangle Packing Problem: RP
Let $\mathcal{M}$ be a set of n rectangular modules of fixed orientations, whose heights and widths are given in real numbers. A *packing* of $\mathcal{M}$ is a non-overlapping placement of the modules. The minimum bounding rectangle of a packing is called the *chip*. Find a packing of $\mathcal{M}$ onto a chip of the minimum area.

A packing of six modules is shown in Fig. 2.1.

The decision version of our problem **RP(A)** is to decide whether $\mathcal{M}$ can be packed onto a chip of area A. Baker, Coffman and Rivest [1] proved the $\mathcal{NP}$-completeness of a similar problem **RP(H,W)** : decide whether $\mathcal{M}$ can be packed onto the chip of height H and width W. We can show **RP(A)** to be $\mathcal{NP}$-complete using the fact that any instance of **RP(H,W)** can be polynomially reducible to an instance of **RP(A)** by the following conversion.

$$
\begin{aligned}
r &\leftarrow \frac{\text{the maximum width over modules}}{2H} \\
A &\leftarrow (W + rH)(W + 2rH) \\
\mathcal{M}' &\leftarrow \{\ (w \times rh)\ \mid\ \forall\,(w \times h) \in \mathcal{M}\} \ \cup\ \{\ rH \times rH\ ,\ (W + rH) \times (W + rH)\ \}
\end{aligned}
$$

Our problem **RP** is harder than **RP(A)**, so $\mathcal{NP}$-hard.

Since the heights and widths of modules are real numbers, **RP** is not simply a combinatorial optimization problem. In fact, there have been several numerical approaches [6, 7]. They first generate a possibly overlapping arrangement of modules, and then move modules to reduce the overlapping cost. But the overlap elimination is very hard for the numerical approaches without ad-hoc post-processing.

An alternative approach is "combinatorial search". In this approach, a set of codes is defined as a *solution space*. Each code represents a construction of placement. A code is

said to be *feasible* if the construction is consistent, i.e. there exists a packing corresponding to the code. The evaluation of a feasible code is the area of the chip, and the evaluation of an infeasible code is infinitely negative. The combinatorial search aims at finding a best code in the solution space. However, exhaustive search of the whole space will take too much time. Since the problem is **NP**-hard, the size of any such solution space is expected to be exponential. Several heuristics have been proposed to find a good solution in a moderate time, for example, simulated annealing and genetic algorithms. Given a time limit, such a heuristic stops the search half-way and outputs the best solution found so far. For this search to be effective, the minimum requirement of the solution space is the following four items.

(1) The solution space is finite.

(2) Every solution is feasible.

(3) Realization of a code is possible in polynomial time.

(4) There exists a code which corresponds to one of the optimal solutions.

The solution space that satisfies the above four requirements is called *P-admissible*.

The reasons for (1),(3) and (4) are obvious. That for (2) is: most heuristics pick up one solution after another along the neighboring structure defined on the space, consulting with the difference of evaluations (gain) to the previous solution. Therefore, if infeasible solutions are included, the continuity will be destroyed and convergence to a feasible solution is not guaranteed.

A known practical solution space is one derived from the *slicing floorplan* proposed by Otten [10] and others. It satisfies (1), (2) and (3). Several optimization heuristics are applied for the space, and one of the most successful approaches uses simulated annealing [8]. However, since the optimal solution can be non-slicing, (4) is not satisfied. This fact discourages us to start searching for the best in the space. Efforts have been paid to let the space include non-slicing structures [12, 11], but they have not been successful to satisfy (4). (Still, a merit of the slicing structure is in the channel routing stage [16].)

Another approach is proposed by Onodera, Taniguchi and Tamaru [13]. They construct a solution space by assigning one out of the four relations, "left of", "right of", "above", and "below", to every pair of modules. This space satisfies (4) since any packing satisfies a combination of the relations. But there are many infeasible codes such as; module a is left of module b, b is left of c and c is left of a. Thus their space is not P-admissible either. As a consequence, the space does not admit heuristics such as simulated annealing. In their paper, exhaustive search with a branch-and-bound technique is applied to find an exactly optimal solution, but the size of tractable problems is limited up to six modules.

This chapter provides a P-admissible solution space, in which each code is a pair of module name sequences. By searching this space, it has become possible to pack hundreds of modules efficiently, as demonstrated in Fig. 2.9 and Fig. 2.10.

To utilize this solution space of **RP** for VLSI layout design, the evaluation of a packing has to be modified to consider wires. Some evaluating functions are available for estimating the final chip area [8, 13]. Among them, we use the formula proposed in [13]. The largest MCNC building-block benchmark was successfully placed by simulated annealing in about 30 minutes (Fig. 2.11).

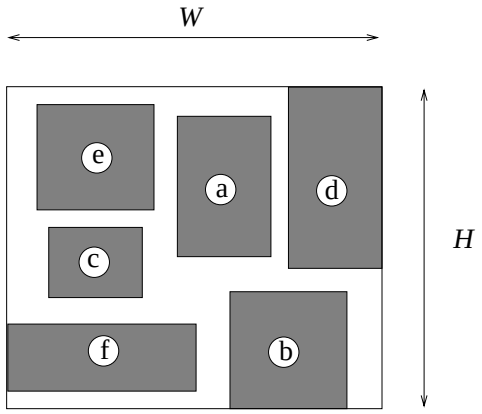


Figure 2.1: A packing on a chip of area $H \times W$

This chapter is organized as follows. In Section 2.2, a mapping from a given packing to a pair of module name sequences is given. It is proved that at least one of the optimal solutions is included in the space. Section 2.3 provides a procedure for an inverse mapping from a sequence pair to a packing. Section 2.4 demonstrates how the space can be utilized in VLSI placement problems. Section 2.5 then concludes with final remarks.

2.2 From Packing to Sequence-Pair

Let Π be a packing on chip C. See Fig. 2.1 for an example. We describe a procedure called **Gridding**, which encodes Π to a *sequence-pair*, an ordered pair of module name sequences.

2.2.1 Gridding

A *rectangular dissection* is a partition of C into rectangles, called *rooms*, such that a room contains at most one module. A room which contains no module is said to be *empty*.

The line segments forming the room boundaries (including four sides of C) are called the *cutting-segs*. We assume that a cutting-seg, except for four sides of C, stops at an inside point of an orthogonal cutting-seg (forming a T-intersection). It is trivial that such a rectangular dissection always exists.

In the following, we describe a procedure to get a pair of module name sequences from a packing.

procedure: Gridding(Π)

Obtain one arbitrary rectangular dissection and fix it. (See Fig.2.2 which is an example rectangular dissection corresponding to Π in Fig.2.1.) Take a

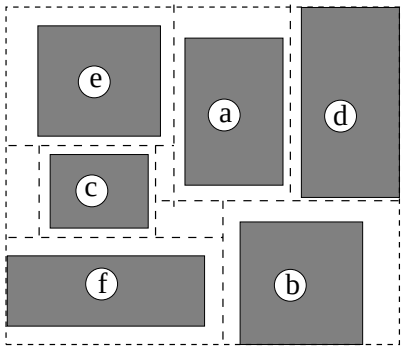


Figure 2.2: A rectangular dissection of a packing

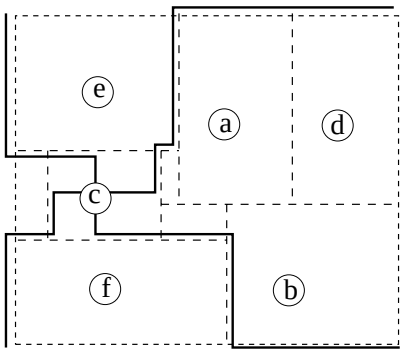


Figure 2.3: Loci of module c

non-empty room. Put a pebble p at the center of the room. Move it right up to hit the cutting-seg which is the side of the room. Then, move p upward until to hit an orthogonal cutting-seg. Then, move it right to hit an orthogonal cutting-seg, and continue turning its direction as right, up, right, up, $\cdots$, until to reach the upper right corner of the chip. The locus of pebble p is called the *right-up locus* of the module. Similarly, *up-left locus*, *left-down locus*, and *down-right locus* are defined. (Fig.2.3 shows these four loci of one module.)

The union of right-up locus of x and left-down locus of x is called the *positive locus* (since it tends to go inside the 1st and 3rd quadrants). Analogously, the union of the up-left locus of x and down-right locus of x is called the *negative locus*. For every module, one positive locus and one negative locus are uniquely defined. They are referred to by the corresponding module names. (An example with all loci is shown in Fig.2.4.)

Theorem 1 :

No pair of positive loci crosses each other. No pair of negative loci crosses each other. (They may run along the same cutting-segs, but not cross each other.)

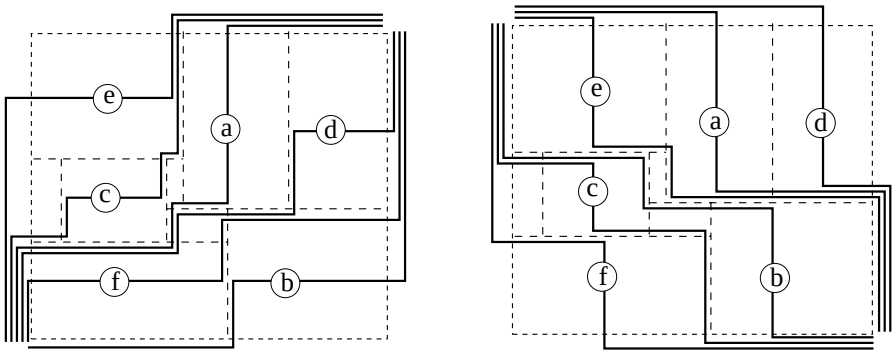


Figure 2.4: Positive loci (left) and negative loci (right), resulted in $(\Gamma_+, \Gamma_-) = (ecadfb, fcbead)$

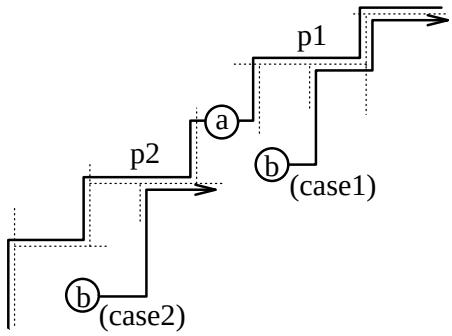


Figure 2.5: Loci used in the proof of Theorem 1

Proof: Let two modules be a and b. Since positive loci of a and b cannot be inside the other room, a crossing, if any, would occur outside their rooms. Denote the right-up locus of module a be RU(a). Similar notation is applied for the other three types loci.

Suppose that RU(b) comes from below and hits RU(a) at a point p_1. See Fig.2.5 case 1. Since RU(a) and RU(b) are along cutting-segs, RU(b) cannot cross RU(a) at p_1 by definition of the cutting-seg. After p_1, the two must run for a while. Since they are following the same rule of right-up locus, they run together and never cross each other. Hence, right-up loci of a and b do not cross. By the same reason, left-down loci of a and b do not cross.

Suppose that RU(b) comes from below and hits LD(a) at a point p_2. See Fig.2.5 case 2. After p_2, RU(b) goes right upstream along LD(a) for a while. Then RU(b) reaches to the point where LD(a) comes from above. After that point, RU(b) continues to go right and thus goes below of LD(a) again. Since RU(b) can not go inside the room of a, it goes below of the room of a. Hence, left-down locus of a and right-up locus of b do not cross. By the same reason, right-up locus of a and left-down locus of b do not cross.

Then, the positive loci of a and b do not cross. Similarly, negative loci of a and b do not cross. □

The implication of the theorem is significant: n positive loci are linearly ordered, and

so are negative loci. Here we order the positive loci from upper left, and order the negative loci from lower left. Since each locus is uniquely referred to by the module name, we have obtained an ordered pair of module name sequences (Γ_+, Γ_-), which we call *sequence-pair*, where Γ_+(resp. Γ_-) is a module name sequence which represents the order of positive (resp. negative) loci.

In Fig.2.4, positive loci are in order "*ecadfb*" and negative loci are in order "$fcbead$", then $(\Gamma_+, \Gamma_-) = (ecadfb, fcbead)$ is obtained.

Given packing Π, the resultant (Γ_+, Γ_-) obtained by **Gridding** is denoted **Gridding**(Π).

2.2.2 Geometrical Information of Sequence-Pair

Let (Γ_+, Γ_-) be the sequence-pair produced by **Gridding** for a packing Π. Modules x and x' are related in exactly one of four ways: x' is after/before x in Γ_+/Γ_-. Let us define four disjoint subsets of $\mathcal{M}$, accordingly.

$$\mathcal{M}^{\mathrm{aa}}(x) = \{x' \mid x' \text{ is after } x \text{ in both } \Gamma_+ \text{ and } \Gamma_-\},$$
$$\mathcal{M}^{\mathrm{bb}}(x) = \{x' \mid x' \text{ is before } x \text{ in both } \Gamma_+ \text{ and } \Gamma_-\},$$
$$\mathcal{M}^{\mathrm{ba}}(x) = \{x' \mid x' \text{ is before } x \text{ in } \Gamma_+ \text{ and after } x \text{ in } \Gamma_-\},$$
$$\mathcal{M}^{\mathrm{ab}}(x) = \{x' \mid x' \text{ is after } x \text{ in } \Gamma_+ \text{ and before } x \text{ in } \Gamma_-\}.$$

For example, with respect to the sequence-pair $(\Gamma_+, \Gamma_-) = (ecadfb, fcbead)$, four subsets for module c are: $\mathcal{M}^{\mathrm{aa}}(c) = \{a, b, d\}$, $\mathcal{M}^{\mathrm{bb}}(c) = \emptyset$, $\mathcal{M}^{\mathrm{ba}}(c) = \{e\}$, and $\mathcal{M}^{\mathrm{ab}}(c) = \{f\}$.

Any module other than x belongs to a unique subset, and it is trivial that two modules are in a dual relation through $x \leftrightarrow x'$, and $a \leftrightarrow b$ as:

$$x' \in \mathcal{M}^{\mathrm{aa}}(x) \Leftrightarrow x \in \mathcal{M}^{\mathrm{bb}}(x')$$
$$x' \in \mathcal{M}^{\mathrm{ba}}(x) \Leftrightarrow x \in \mathcal{M}^{\mathrm{ab}}(x')$$

In a packing, if the right side of module x is left of the left side of module x', x is said to be *left of* x'. Similarly, *right of*, *above*, *below* relations between two modules are defined.

Theorem 2 :
Let (Γ_+, Γ_-) be the sequence-pair produced by **Gridding** for a packing Π. If $x \in \mathcal{M}^{\mathrm{bb}}(x')$, then x is left of x' in Π. The claim also holds when the pair of words ("$\mathcal{M}^{\mathrm{bb}}$" and "left of") is replaced by any of ("$\mathcal{M}^{\mathrm{aa}}$" and "right of"), ("$\mathcal{M}^{\mathrm{ba}}$" and "above"), and ("$\mathcal{M}^{\mathrm{ab}}$" and "below").

In the previous example, module b is in $\mathcal{M}^{\mathrm{aa}}(c)$. One can examine b is actually right of c in the packing shown in Fig. 2.1.

Proof: Let x and x' be arbitrary two modules. The loci of x divide the chip into four regions. Among them, the region surrounded by the up-right locus of x and right-down locus of x together with the right side of the chip is called the *right-cone* of x. Analogously, the *left-*, *above-*, and *below-cone* denote the other three regions.

Suppose x' is in $\mathcal{M}^{\mathrm{aa}}(x)$. This implies that the positive locus of x' is in the union of the right-cone and the below-cone of x. Also it is implied that the negative locus of x' is in the union of the right-cone and the above-cone of x. The cross point of the positive locus and the negative locus of x' is in their intersection, that is, the right-cone of x. Then, module x' is in the right-cone of x. Every modules in the right-cone of x is right of module x by definition of the up-right locus and the right-down locus of x.

It is clear that the claim holds for the other cases. □

2.3 From Sequence-Pair to Packing

In the previous section, we analyzed the packing and fixed the procedure **Gridding** to obtain one sequence-pair from a given packing. Now we provide a procedure to synthesize one packing from an arbitrary sequence-pair.

2.3.1 Constraint of Sequence-Pair

Given a sequence-pair (Γ_+, Γ_-), we read a constraint from it as follows.

The Constraint Implied by a Sequence-Pair (Γ_+, Γ_-)
If $x \in \mathcal{M}^{\mathrm{bb}}(x')$, module x must be left of module x'. This is also the constraint with replacing the pair of words ("$\mathcal{M}^{\mathrm{bb}}$" and "left of") with any of ("$\mathcal{M}^{\mathrm{aa}}$" and "right of"), ("$\mathcal{M}^{\mathrm{ba}}$" and "above"), and ("$\mathcal{M}^{\mathrm{ab}}$" and "below").

It is easily seen that the constraint imposed on the packing by a sequence-pair is unique. Furthermore, the following theorem holds.

Theorem 3 : The constraint is always satisfiable.

Proof: Consider an $n \times n$ grid. Label the horizontal grid lines and vertical grid lines with module names along Γ_+ and Γ_- from top and from left, respectively. A cross point of the horizontal grid line of label x and the vertical grid line of label x' is referred to by (x, x'). Then, rotate the resultant grid by 45 degrees counter clockwise to get an oblique grid. (See Fig. 2.6.) Put each module x with its center being on (x, x). Expand the separation of grid lines $\sqrt{2}$ times larger than the longest width/height over modules, which is sufficient to eliminate overlapping of modules. The resultant packing trivially satisfies the constraint implied by the given sequence-pair. □

An example is shown in Fig. 2.6.

2.3.2 Best Packing Under the Constraint

Given (Γ_+, Γ_-), one of the optimal packings under the constraint can be obtained in $O(n^2)$ time by applying the well-known *longest path algorithm* for vertex weighted directed acyclic graphs. The process is given below.

Based on "left of" constraint of (Γ_+, Γ_-), a directed and vertex-weighted graph $G_H(V, E)$ (V: vertex set, E: edge set), called the *horizontal-constraint graph*, is constructed as follows.

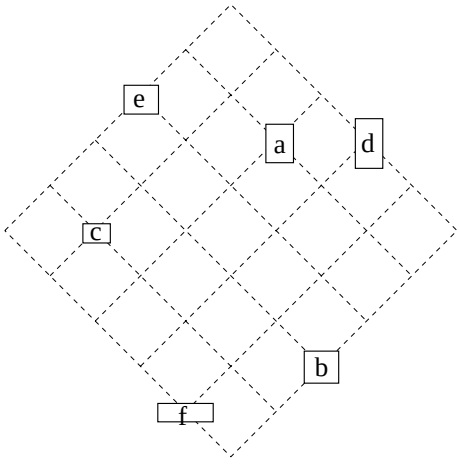


Figure 2.6: A packing on an oblique grid for $(\Gamma_+, \Gamma_-) = (ecadfb, fcbead)$

V : source s, sink t, and n vertices labeled with module names

E : (s, x) and (x, t) for each module x, and (x, x') if and only if $x \in \mathcal{M}^{\mathrm{bb}}(x')$ ("left of" constraint)

Vertex-weight : zero for s and t, width of the corresponding module for the other vertices

Similarly the *vertical-constraint graph* $G_V(V, E)$ is constructed using "below" constraint and the height of each module.

Neither of these graphs contains any directed cycle. We set the X-coordinate of x to be the longest path length from s to x in G_H. The Y-coordinate of x is set independently using G_V. If two modules x and x' are in horizontal relation, then there is an edge between x and x' in G_H, hence they do not overlap horizontally in the resultant placement. Similarly, if x and x' are in vertical relation, they do not overlap vertically. Thus no two modules overlap each other in the resultant placement because any pair of modules is either in horizontal or vertical relation.

The width and the height of the chip is determined by the longest path length between the source and the sink in G_H and G_V, respectively. Since the width and the height of the chip is independently minimum, the resultant packing is the best of all the packings under the constraint. The longest path length calculation on each graph can be done in $O(n^2)$ time, proportional to the number of edges in the graph.

As an example, G_H and G_V are shown in Fig. 2.7 for $(\Gamma_+, \Gamma_-) = (ecadfb, fcbead)$. The resultant placement after the longest path length calculation is shown in Fig. 2.8.

2.3.3 P-admissible Solution Space

Previous discussions conclude:

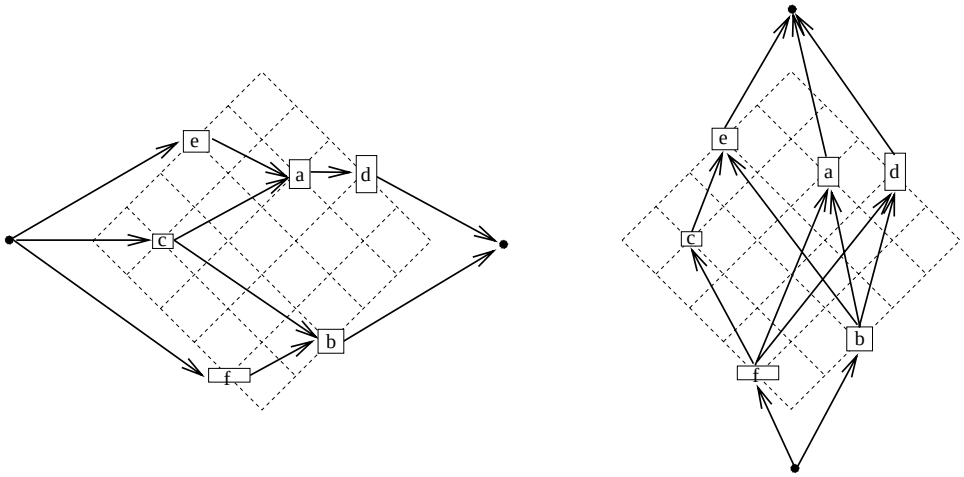


Figure 2.7: Constraint graphs G_H(left) and G_V(right) (transitive edges are not drawn for simplicity)

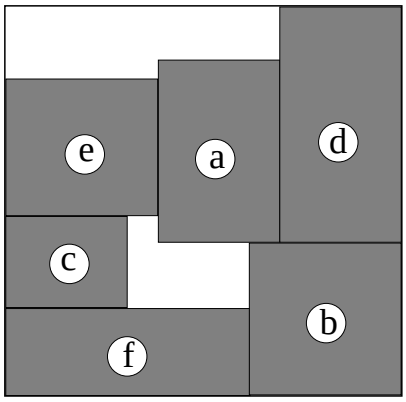


Figure 2.8: A best packing under the constraint implied by $(\Gamma_+, \Gamma_-) = (ecadfb, fcbead)$

Theorem 4 : The set of all sequence-pairs is a P-admissible solution space of **RP**. More precisely, it consists of $(n!)^2$ sequence-pairs, each of which can be mapped to a packing in $O(n^2)$ time, and at least one of which corresponds to one of the optimal solutions of **RP**. □

Our discussion started for minimizing the area of the chip. However, all the discussions hold as long as the evaluating function is independently non-decreasing with respect to the width and the height of the chip. Therefore we may assume instead, for example, perimeter of the chip, area of the chip of pre-specified aspect ratio, and the height of the chip when its width is fixed. This fact will extend the usefulness of our solution space.

It has also been assumed that the orientation of each module (vertically laid or horizontally laid) is fixed. When the orientation is also requested to be optimized, we hold a $\{0, 1\}$ sequence of length n, expressing the orientation of each module being horizontal or vertical. The size of solution space increases to $(n!)^2 2^m$. (The orientation optimization for a fixed rectangular dissection is known to be NP-hard [17].) This technique can be easily extended to so-called "soft" modules, by preparing three or more candidates of (width, height) per module [18].

There is a sequence-pair for which another sequence-pair provides no worse packing, independent of the sizes of the modules. For example, if $(abcd, cdab)$ corresponds to an optimal packing, then $(abcd, cadb)$ or $(acbd, cdab)$ also corresponds to an optimal packing, regardless of the widths and the heights of modules a, b, c and d. Then, the former sequence-pair, $(abcd, cdab)$, is redundant for our current objective to find a packing with smaller area. We extend our evaluating function to consider wires in the next section.

2.4 Use of Sequence-Pair

2.4.1 Rectangle Packing

We use a standard simulated annealing method to pack rectangles. It uses two kinds of pair-interchanges: (i) two module names in Γ_+, (ii) two module names in Γ_+ and also in Γ_-. The initial sequence-pair was made at random. The temperature was decreased exponentially.

The first interest would be to know the experimental performance ratio (obtained area / optimal area) of the above described simulated annealing. For this purpose, two problem instances are constructed such that their optimal solutions are known.

REGGRID : a collection of 100 unit squares. It is easily understood that an optimal solution with area 100 is possible when the squares are packed in the 10×10 regular grid.

LOGGRID : a collection of 100 rectangles, generated by an exponential grid formed by eleven vertical lines $x = 0, 1, 2, 3, 5, 7, 10, 14, 19, 26, 36$ and eleven horizontal lines $y = 0, 1, 2, 3, 5, 7, 10, 14, 19, 26, 36$. Thus, the area of the optimal solution is $36 \times 36 = 1296$.

The result is shown in Table 2.1. The calculation speed was $0.78 \sim 0.80$ milliseconds per iteration on a Sun SS-5 with 75 MHz clock. From Table 2.1, the proposed method attained the area 1.2 times larger than the minimum in a moderate time.

Number of iterations	**REGGRID**	**LOGGRID**
10000	1.44	1.29
30000	1.32	1.33
100000	1.21	1.17
300000	1.3	1.17
1000000	1.08	1.20
3000000	1.20	1.14
10000000	1.1	1.11
30000000	1.1	1.14

Table 2.1: Performance ratio of 100 modules packing

To know the performance of the proposed method for "real" data, dimensions of 146 modules were extracted from a printed circuit board in a personal computer. The simulated annealing process is designed similarly, but we also include orientation optimization for this example: (i) two module names in Γ_+, (ii) two module names in Γ_+ and also in Γ_-, (iii) the width and the height of a module, where the last one is for orientation optimization. The initial sequence-pair was made as $\Gamma_+ = \Gamma_-$, which corresponds to a linear horizontal arrangement of modules. From a heuristic point of view, operation (i) was selected with higher probability in higher temperature, and operation (iii) was selected with higher probability in lower temperature.

The result is shown in Fig. 2.9. Computation on Sun SS-2 stopped in 29.9 minutes reaching the terminating temperature. The algorithm searched at most 606,192 distinct sequence-pairs out of the solution space of size $(146!)^2 2^{146} \sim 1.23 \times 10^{552}$. Notice that, the search of only a fraction about 4.92×10^{-547} of the solution space was enough to obtain the placement shown in Fig. 2.9.

As another challenge, we tried 500 modules, using 18.83 hours on a Sun SS-2 to get the result shown in Fig. 2.10.

2.4.2 Module Placement with Wires

For VLSI placement, we extend the evaluation to consider wires. Among various possible evaluations about wires, we focus on the final chip area after all wires are detailed-routed. However, it is difficult at this moment and hence we made use of an estimate of the final chip area without the actual routing phase.

Let (Γ_+, Γ_-) be a sequence-pair, Π be one of the optimal packings under the constraint implied by (Γ_+, Γ_-), and W and H be the width and the height of Π. Terminals are given as fixed points on the boundary of each module. A *net* is a set of terminals (multi-terminal net), which must be connected by wires, later in detailed-routing phase. A set of nets is given as the netlist $\mathcal{N}$. Given terminals of a net i, which spread over Π, the width and the height of the smallest bounding box of these terminals are denoted by W_i and H_i, respectively. T is the sum of wiring width and wiring space (obtained from a design rule set). We use the following formula proposed in [13] to estimate the final chip width W' and height H'.

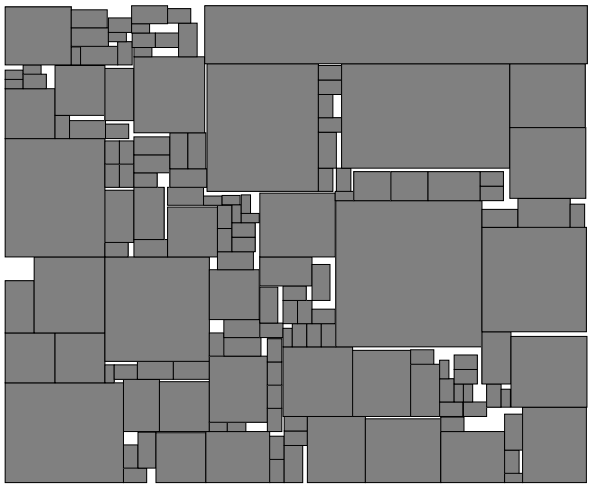


Figure 2.9: Packing of 146 modules

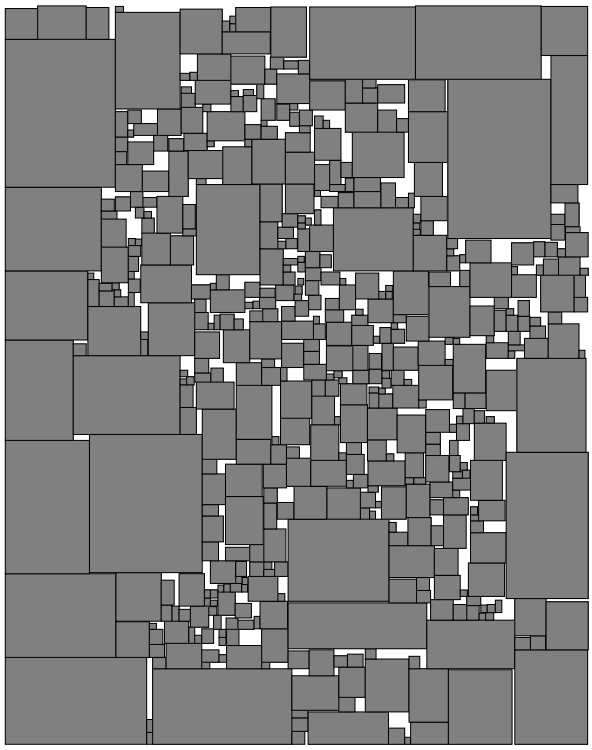


Figure 2.10: Packing of 500 modules

$$W' = W + T\frac{\Sigma_{i\in\mathcal{N}}H_i}{H}$$
$$H' = H + T\frac{\Sigma_{i\in\mathcal{N}}W_i}{W}$$

The second term of each formula estimates the increase in one direction owing to the wires, assuming all wires are uniformly distributed in the final chip. They experimentally showed that the result is acceptable for a commercial channel router [13].

There are choices per module, which is the combination of the four choices of 0, 90, 180, 270 degree rotations, and a decision *yes, no* on reflecting the module about the Y axis. This code for orientation and a sequence-pair are put together into a simulated annealing process in our system. The process runs in a similar fashion as the rectangle packing optimization, and explores the solution space of size $(n!)^2 8^n$.

A point not mentioned in [13] is how the location of each individual module is calculated. In our system, after the best evaluated code is obtained, coordinates of each module are determined as follows. Assume (X_j, Y_j) is the coordinates of the lower left corner of module j in Π. (This is the information we can use in this phase.) Let $\mathcal{N}_{X_j}$ be a set of nets such that the X coordinate of the left side of bounding box of the net is less than or equal to X_j. Similarly, $\mathcal{N}_{Y_j}$ is defined using Y_j. We determine the coordinates (X'_j, Y'_j) of the lower left corner of module j in the resultant chip by the following formula.

$$X'_j = X_j + T\frac{\Sigma_{i\in\mathcal{N}_{X_j}}H_i}{H}$$
$$Y'_j = Y_j + T\frac{\Sigma_{i\in\mathcal{N}_{Y_j}}W_j}{W}$$

For an experiment, the biggest building block layout data, called "ami49", was taken from the MCNC benchmarks. The data is the biggest one in their benchmark suit, but our method was fast enough to handle the data without splitting the problem. (Some recent research [9] also handle the data without dividing the problem.) The result is shown in Fig. 2.11. We remark that it was done with an additional constraint: aspect ratio = 1, taking $T = 7\ \mu m$. The estimated chip size is 6482 μm × 6925 μm. Computation time was 31.36 minutes on SunIPX.

2.5 Conclusion

This chapter introduced a data structure to represent a general packing in terms of a pair of module name sequences, called sequence-pair. Detailed proofs are presented to show that every sequence-pair feasibly corresponds to a packing, and at least one sequence-pair corresponds to an area-minimum packing.

In experiments, 500 rectangles were packed very efficiently in a reasonable time. It was attained by a standard simulated annealing in which a move is a change of the sequence-pair. The evaluating function was then extended to the VLSI placement problem using a conventional wiring area estimation method. The biggest MCNC benchmark, ami49, is placed very nicely.

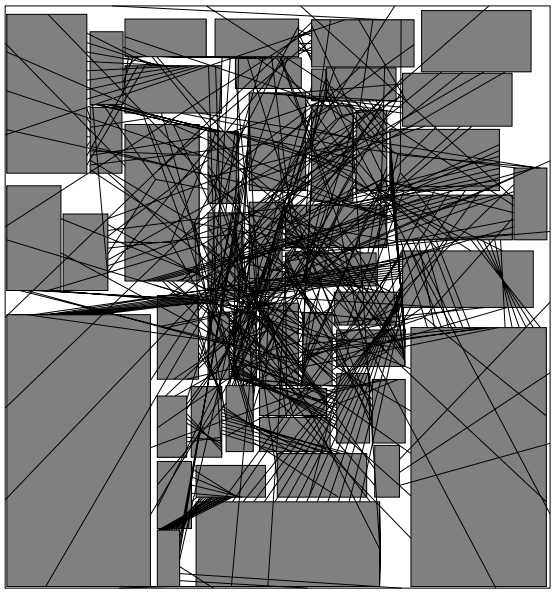


Figure 2.11: Placement of MCNC “ami49”

Chapter 3

Mapping from Sequence-Pair to Rectangular-Dissection

3.1 Introduction

In the first stage of VLSI physical design, it is required to determine a rough arrangement of circuit components, such as modules and channels. A stochastic algorithm, such as simulated annealing or genetic algorithm, would be a good choice as an optimization algorithm since the problem is hard. To make a stochastic algorithm work effectively, a fundamental issue is in how to represent candidate arrangements, with enough generality and efficiency to cope with various design requirements.

In Chapter 2, we proposed a representation called *sequence-pair*, which is a pair of module name sequences. For example, (abc, cab) is a sequence-pair for module set $\{a, b, c\}$. For a sequence-pair, they assigned an *HV-relation-set (HVRS)*, which is a set of horizontal (right of/left of) or vertical (above/below) relations for every module pair. For example, sequence-pair (abc, cab) corresponds to HVRS $\{a$ is left of b, c is below a, c is below $b\}$. It is proved in the chapter that a sequence-pair always corresponds to a realizable HVRS, and there is a sequence-pair whose HVRS can lead an area minimum placement. However, HVRS alone is not sufficient as a representation of candidate arrangements of components. Channel positions are also desired to be represented together.

A traditional method exists to represent channel positions together with module positions. It is the *rectangular-dissection*. (sometimes called *floorplan* in the literature [17].) However, known efficient representation techniques are limited for specific classes of rectangular-dissections, such as *slicing structure* [8].

To combine the merits of sequence-pair and rectangular-dissection, it is desired to map a sequence-pair to a rectangular-dissection. Observe from Fig. 3.2-(a) that a room with no module assignment, called the *empty room*, is necessary in the rectangular-dissection to keep the relative positions of modules. It is worth allowing this empty room since it is essentially needed to achieve the area minimum placement. However, introducing arbitrary many empty rooms results in arbitrary many line segments, which represent channels.

This chapter gives a mapping from a sequence-pair to a rectangular-dissection whose number of rooms is minimum among all the rectangular-dissections whose HVRSs are equivalent to the HVRS of the given sequence-pair. Consequently, candidate arrangements of modules and channels are successfully represented with the generality and the efficiency

inherited from the sequence-pair.

The organization of this chapter is as follows. Section 3.2 defines preliminary terms. Section 3.3 shows a necessary and sufficient condition that a sequence-pair is mapped to a rectangular-dissection with no empty room. Section 3.4 presents a procedure to output a rectangular-dissection with fewest empty rooms. Section 3.5 is for conclusion.

3.2 Preliminary

3.2.1 HV-Relation-Set

An *HV-relation-set* for a set of modules is a set of horizontal (right of / left of) or vertical (above/below) relations for all module pairs. For example,

$$\{a \text{ is left of } b,\ c \text{ is below } a,\ c \text{ is below } b\}$$

is an HVRS for module set $\{a, b, c\}$. The cardinality of an HVRS is $\binom{n}{2}$, where n is the number of modules. The variety of HVRS is $4^{\binom{n}{2}}$.

An HVRS may or may not be realizable. The above example is realizable. A non-realizable example is : $\{a$ is left of b, b is left of c, c is left of $a\}$. A branch and bound approach [13] can be used to eliminate non-realizable HVRSs.

3.2.2 Sequence-Pair

A *sequence-pair* is an ordered pair of Γ_+ and Γ_-, where each of Γ_+ and Γ_- is a sequence of names of given n modules. For example, $(\Gamma_+, \Gamma_-) = (abcd, bdac)$ is a sequence-pair of module set $\{a, b, c, d\}$. If module x is the i'th module in Γ_+, we denote $\Gamma_+(i) = x$, as well as $\Gamma_+^{-1}(x) = i$. A similar notation is used also for Γ_-. To help intuitive understanding, we use a notation such as

$$(\Gamma_+, \Gamma_-) = (\cdots a \cdots b \cdots, \cdots a \cdots b \cdots)$$

by which we mean

$$\Gamma_+^{-1}(a) < \Gamma_+^{-1}(b) \quad \text{and} \quad \Gamma_-^{-1}(a) < \Gamma_-^{-1}(b).$$

A sequence-pair corresponds to an HVRS as follows. For every module pair $\{a, b\}$, a is left of b (equivalently, b is right of a) if

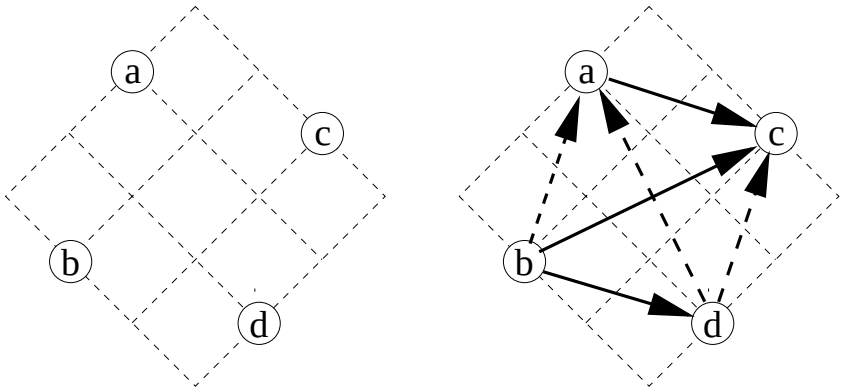
$$(\Gamma_+, \Gamma_-) = (\cdots a \cdots b \cdots, \cdots a \cdots b \cdots).$$

Similarly, a is below b (equivalently, b is above a) if

$$(\Gamma_+, \Gamma_-) = (\cdots b \cdots a \cdots, \cdots a \cdots b \cdots).$$

For example, sequence-pair $(abcd, bdac)$ implies HVRS: $\{b$ is below a, b is left of d, d is below c, a is left of c, d is below a, b is left of $c\}$.

The variety of HVRS represented by the sequence-pair equals to the variety of the sequence-pair, $(n!)^2$, thus drastically reduced from the original variety $4^{\binom{n}{2}}$, where n is the number of modules. Furthermore, the sequence-pair has the following property.



(a) (b)

Figure 3.1: (a) Sequence-Pair $(abcd, bdac)$, (b) Horizontal-Seq-Pair-Graph (H-SPG) and Vertical-Seq-Pair-Graph (V-SPG). The edges of H-SPG are drawn in solid lines and the edges of V-SPG are drawn in dotted lines.

Property 1 : The HVRS of every sequence-pair is realizable. For any non-overlapping placement, there is a sequence-pair whose HVRS is satisfied by the placement. □

A proof is given in Chapter 2.

The HVRS of a sequence-pair of n modules can be graphically understood by means of *oblique-grid*, defined as follows. Let $L_+(1), L_+(2), \cdots, L_+(n)$ be n parallel lines of slope $+1$ drawn on a plane, ordered from left. Let $L_-(1), L_-(2), \cdots, L_-(n)$ be n parallel lines of slope -1 drawn on the plane, also ordered from left. These $2n$ lines form a 45 degree oblique $n \times n$ grid, called the *oblique-gird*. The *oblique-grid-embedding* of a sequence-pair (Γ_+, Γ_-) is the oblique-grid with each module name x written at the cross point of $L_+(\Gamma_+^{-1}(x))$ and $L_-(\Gamma_-^{-1}(x))$. Fig. 3.1-(a) shows the oblique-grid-embedding of sequence-pair $(abcd, bdac)$. Using the oblique-grid-embedding, the HVRS of a sequence-pair can be re-defined as: for each module x, the modules which are seen from x in the angle between -45 degree and 45 degree are right of x, the modules in the angle between 45 degree and 135 degree are above x, and so on.

The HVRS of a sequence-pair is represented by a pair of directed acyclic graphs, called *horizontal-sequence-pair-graph (H-SPG)* and *vertical-sequence-pair-graph (V-SPG)*, defined as follows. For either graph, vertices uniquely correspond to modules and have the corresponding module names. The edge set of the H-SPG is constructed faithfully to the horizontal relations, from left to right, but eliminating the transitive edges. The edge set of the V-SPG is defined similarly from bottom to top. We sometime abbreviate the pair of H-SPG and V-SPG of a sequence-pair to "SPGs".

Oblique-grid-embedding of a sequence-pair with arrows additionally drawn corresponding to the edges of the SPGs is called the oblique-grid-embedding of the SPGs. Fig. 3.1-(b) shows an example, where the edges of the H-SPG are drawn using solid lines, and the edges of the V-SPG are drawn using dotted lines.

3.2.3 Rectangular-Dissection

A *rectangular-dissection* is a dissection of a rectangle into a set of rectangles, called *rooms*, with an injective assignment of modules to rooms (no two modules share a room.) An example is shown in Fig. 3.2-(a). Only T-intersections are used to form the dissection except for the four corners of the bounding rectangle. (Two T-intersections may form a cross shape as a degenerate case.) The bounding rectangle represents the chip, each room represents an area which is assignable to a module, and each line segment represents a channel. A room is said to be *occupied* if a module is assigned to the room, otherwise said to be *empty*. In Fig. 3.2-(a), the dark room at the center is empty and the other rooms are occupied. Empty rooms have been used to modify a rectangular-dissection incrementally [19].

A rectangular-dissection specifies relative positions of modules and channels as follows: If the right side of a room r_a and the left side of a room r_b are both on an identical vertical line segment l_c, the module a assigned to the room r_a should be placed left of the channel c corresponding to the line segment l_c, and the module b assigned to the room r_b should be placed right of the channel c (horizontal relation). Notice that a horizontal relation between module pair a, b is transitively specified as: module a should be placed left of module b. Vertical relations are specified similarly using horizontal line segments.

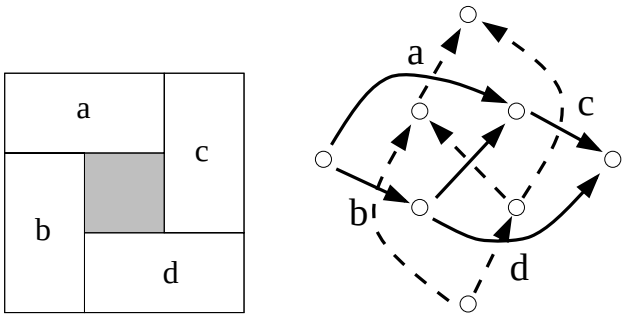
The information of a rectangular-dissection is commonly represented by means of a pair of directed acyclic graphs [20, 21, 17], a *horizontal-rectangular-dissection-graph (H-RDG)* and a *vertical-rectangular-dissection-graph (V-RDG)*. Each vertical (horizontal) line segment corresponds to a vertex in the H-RDG (V-RDG) and each room corresponds to an edge (u, v) where u is the vertex corresponding to the left (bottom) side of the room and v is the vertex corresponding to the right (top) side of the room. We sometime use the word "RDGs" to denote the pair of H-RDG and V-RDG of a rectangular-dissection. Two rectangular-dissections are said to be equivalent if their RDGs (the two H-RDGs, as well as the two V-RDGs) are the same. Fig. 3.2-(b) illustrates the RDGs of the rectangular-dissection shown in Fig. 3.2-(a). In the figure, the edges of H-RDG are drawn using solid lines, and the edges of V-RDG are drawn using dotted lines. An empty room corresponds to the anonymous edge in the figure.

H-RDG as well as V-RDG is a directed acyclic planar graph with possibly duplicated edges. Each RDG is polar, i.e. a directed acyclic graph with a single source and a single sink. Two polar graphs G_1 and G_2 are said to be in *polar-dual* relation if G_1 and G_2 become dual when an undirected edge from the source to the sink is added in each graph. From the construction, the RDGs are in polar-dual relation. The reverse is also true since polar-dual graphs are known to be mapped to a rectangular-dissection [21].

Property 2 : Given two polar graphs G_1 and G_2, there exists a rectangular-dissection whose RDGs are G_1 and G_2, if and only if G_1 and G_2 are in polar-dual relation. □

When we construct a rectangular-dissection from H-RDG G_h and V-RDG G_v, we use the following procedure.

Procedure ConstRD(G_h, G_v)
For a vertex $u \in V(G_h)$, $x(u)$ denotes the ordinal number of the vertex u in a topological order of the vertices in G_h. ($x(u)$ has a unique integer such that $x(u) < x(u')$ if there exists a path from u to u'). Similarly, $y(v)$ denotes the ordinal number of the vertex v



(a) (b)

Figure 3.2: (a) Rectangular-Dissection, (b) Horizontal-Rectangular-Dissection-Graph (H-RDG) and Vertical-Rectangular-Dissection-Graph (V-RDG). H-RDG is drawn in solid lines and V-RDG is drawn in dotted lines.

in a topological order of the vertices in G_v. A pair of edges (e_h, e_v) is called a "cross" if $e_h (\in E(G_h))$ and $e_v (\in E(G_v))$ are in a dual relation. For each cross $((u_1, u_2), (v_1, v_2))$, draw a rectangle whose lower left corner is at $(x(u_1), y(v_1))$ and whose upper right corner is at $(x(u_2), y(v_2))$. (**Procedure** ConstRD **End**)

It is easily seen that ConstRD runs in $O(n)$ time, where $n = |E(G_h)| = |E(G_v)|$ which also equals to the number of rooms in the resultant rectangular-dissection.

3.2.4 Sequence-Pair and Rectangular-Dissection

The major merit of the sequence-pair and that of the rectangular-dissection are summarized as follows.

- The merit of the sequence-pair is in its efficiency in enumerating various HVRSs.
- The merit of the rectangular-dissection is in its ability of representing the channels.

To keep the two merits at the same time, the target of this chapter is:

Target: To map a sequence-pair to a rectangular-dissection.

The following three properties show a similarity of the sequence-pair and the rectangular-dissection.

Property 3 : Given a sequence-pair, for any two modules a and b, there is a path which connects a and b in H-SPG or in V-SPG, but not in both. □

Property 4 : In the H-RDG (V-RDG) of a rectangular-dissection, if there is a path from edge a to edge b, then the room a is left of (below) room b in the rectangular-dissection. □

Property 3 and 4 are easily understood.

Property 5 : Given a rectangular-dissection, for any two rooms a, b, there is a path which connects a and b in H-RDG or in V-RDG, and not in both.
Proof : Let G and G' be the H-RDG and V-RDG of the rectangular-dissection. Then G and G' are in polar-dual relation. Let source and sink of $G(G')$ be $s(s')$ and $t(t')$, respectively. A *full-path* of $G(G')$ is a path from $s(s')$ to $t(t')$ in $G(G')$. Then the claim can be re-written as: for any two edges a and b, a full-path which includes both a and b exists either in G or G', and not exists in both G and G'.

It is clear that G and G' are in polar-dual relation. Hence, the edge set of a full-path of $G(G')$ has one to one correspondence with a cut set of $G'(G)$.

If G has a full-path which includes both a and b, there is a cut set in G' which includes both a and b, hence G' does not have a full-path which includes both a and b.

In the following, we consider the case G does not have a full-path which includes both a and b. Let V_R be the subset of vertices in G consists of the vertices which is reachable from the outgoing vertex of a or the outgoing vertex of b. Let $V_{\overline{R}}$ be the rest. Since G is a directed acyclic graph, the incoming vertex of G and the incoming vertex of G' are both in $V_{\overline{R}}$. There is no edge from a vertex in V_R to a vertex in $V_{\overline{R}}$, hence the set of edges from a vertex in $V_{\overline{R}}$ to a vertex in V_R is a cut, and the cut includes both a and b. Therefore, G' has a full-path which includes both a and b. □

Property 4 and 5 imply that a rectangular-dissection, as well as a sequence-pair, uniquely corresponds to an HVRS. Then, the correspondence between the sequence-pair and the rectangular-dissection is in question. Next property can be easily derived from the result of Chapter 2.

Property 6 : For the HVRS T of any rectangular-dissection, there is unique sequence-pair S whose HVRS is T. □

The reverse direction is essential to achieve our target. We have the following observations.

Observation 1 : There is a sequence-pair whose HVRS can only be represented by a rectangular-dissection with an empty room. □

$(abcd, bdac)$ is an example of such sequence-pair whose HVRS can only be represented using an empty room. Fig. 3.1-(a) and Fig. 3.2-(a) illustrate the sequence-pair and the corresponding rectangular-dissection.

Observation 2 : There is a set of modules whose area minimum placement can only be represented by a rectangular-dissection with an empty room. □

For instance, area minimum placement of four modules of sizes 3×2, 2×3, 3×3 and 2×4, can be represented essentially only by the rectangular-dissection shown in Fig. 3.2-(a). From Observation 1 and 2, it is our constraint that:

Constraint: The HVRS of a sequence-pair should be preserved by the targeted mapping.

Observation 3 : For an HVRS, rectangular-dissection is not unique if arbitrary many empty rooms are allow to be introduced. □

Property 7 : For any rectangular-dissection, the number of line segments is equal to the number of rooms plus three. □

Property 7 can be proved by counting the number of room corners contributed by a line segment.

Recall that the line segments represent channels. Although the goodness about the number of channels might differ in several routing schemes, fewer number of channels is most likely preferred to avoid too many wire bends. Thus, it is our criterion that:

Criterion: Minimize the number of rooms in the targeted mapping.

3.3 Rectangular-Dissection without Empty Room

This section gives a procedure which maps a sequence-pair to a rectangular-dissection without any empty room if the given sequence-pair satisfies a certain condition. Then, the condition is revealed to be necessary and sufficient for eliminating the introduction of empty room. To describe the condition, we need to define two terms, *HV-cross* and *adjacent-cross*.

3.3.1 HV-Cross and Adjacent-Cross

Four modules a, b, c, d are said to form an *HV-cross* in a sequence-pair $S = (\Gamma_+, \Gamma_-)$ if they satisfy the following three conditions in (Γ_+, Γ_-) or in (Γ_+, Γ'_-), where Γ'_- is the reverse of Γ_-.

- $(\cdots a \cdots b \cdots c \cdots d \cdots, \cdots c \cdots a \cdots d \cdots b \cdots)$
- There is no module x which satisfies

$$(\cdots a \cdots x \cdots d \cdots, \cdots a \cdots x \cdots d \cdots).$$

- There is no module x which satisfies

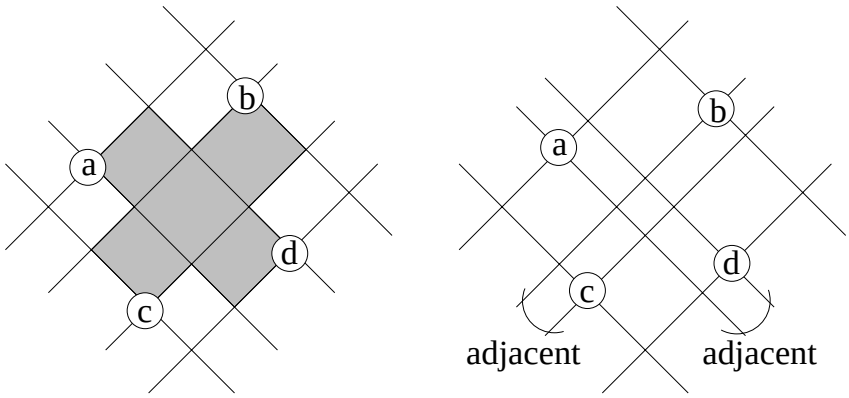
$$(\cdots b \cdots x \cdots c \cdots, \cdots c \cdots x \cdots b \cdots).$$

Fig. 3.3-(a) illustrates an HV-cross using oblique-grid. There is no module in the dark region because of the last two conditions in the definition. HV-cross is so called because it corresponds to a crossing between an edge in the H-SPG and an edge in the V-SPG in the oblique-grid-embedding of the SPGs.

If four modules a, b, c, d form an HV-cross and b and c are adjacent in Γ_+, and a and d are adjacent in $\Gamma_-(\Gamma'_-)$, the HV-cross is also called the *adjacent-cross*. The condition is illustrated in Fig. 3.3-(b).

Lemma 1 : If there is an HV-cross in a sequence-pair S, then an adjacent-cross also exists in S.

Proof : The proof is by contradiction. Without loss of generality, let an HV-cross formed by four modules a, b, c and d be $S = (\Gamma_+, \Gamma_-) = (\cdots a \cdots b \cdots c \cdots d \cdots, \cdots c \cdots a \cdots d \cdots b \cdots)$. (See



(a) (b)

Figure 3.3: (a) HV-cross (no module is in the dark region). (b) adjacent-cross (special case of HV-cross).

Fig. 3.4.) We can assume further that: (i) the distance between b and c in Γ_+ is minimal over all the HV-crosses in S; and (ii) among such HV-crosses, the distance between a and d in Γ_- is minimal.

If b and c are not adjacent in Γ_+, there is a module in between. Such modules are not between b and c in Γ_-, from the definition of HV-cross. In such modules, there is module x which satisfies one of the two cases:

- $S = (\cdot\cdot\, a \cdot\cdot b \cdot\cdot x \cdot\cdot c \cdot\cdot d \cdot\cdot, \cdot\cdot\, x \cdot\cdot c \cdot\cdot a \cdot\cdot d \cdot\cdot b \cdot\cdot)$ and a, b, x, d form an HV-cross, or
- $S = (\cdot\cdot\, a \cdot\cdot b \cdot\cdot x \cdot\cdot c \cdot\cdot d \cdot\cdot, \cdot\cdot\, c \cdot\cdot a \cdot\cdot d \cdot\cdot b \cdot\cdot x \cdot\cdot)$ and a, x, c, d form an HV-cross.

(Fig. 3.4 illustrates an example for the former case.) Either case contradicts to the assumption (i). Similarly, if a and d are not adjacent in Γ_-, a contradiction to the assumption (ii) is derived. Hence, a, b, c, d form an adjacent-cross. □

3.3.2 Converting Sequence-Pair into Rectangular-Dissection

A procedure called SeqPair–RDG is presented to map a sequence-pair to a pair of RDGs. From the resultant RDGs, a rectangular-dissection is obtained by the procedure ConstRD given in Section 3.2. Fig. 3.5 illustrates the result of each step for input sequence-pair $S = (abcde, becad)$. A hyper directed edge is denoted (V_i, V_o), where V_i is the input vertex set, and V_o is the output vertex set.

Procedure SeqPair–RDG

Input: Sequence-pair $S = (\Gamma_+, \Gamma_-)$ which has no adjacent-cross.

Output: H-RDG G_{HP} and V-RDG G_{VP}.

(Step 1) Add four new modules s_h, t_h, s_v, t_v, called *phantom* modules, to the input sequence-pair $S = (\Gamma_+, \Gamma_-)$ and obtain new sequence-pair $S^\star = (t_v s_h \Gamma_+ t_h s_v, s_v s_h \Gamma_- t_h t_v)$. Construct H-SPG G_{HSP} and V-SPG G_{VSP} from $S^\star$.

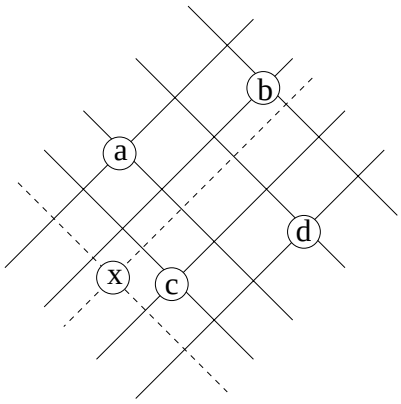


Figure 3.4: Figure used in the proof of Lemma 1

(Step 2) Construct a horizontal hyper graph G_H and a vertical hyper graph G_V from G_{HSP} and G_{VSP} as follows. The vertex set of G_H and G_V are both equivalent to the vertex set of SPGs. A hyper edge (V_L, V_R) is in the edge set $E(G_H)$ if and only if the subgraph of G_{HSP} induced by $V_L \cup V_R$ is a maximal bipartite. The edge set $E(G_V)$ is similarly defined using G_{VSP}.

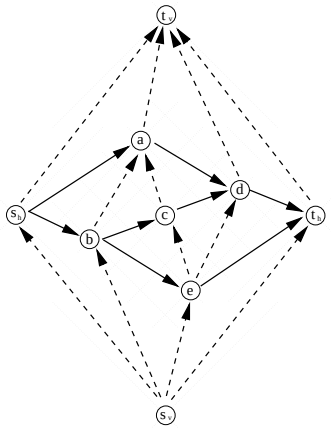
(Step 3) For G_H (also for G_V), construct a hyper graph G_{HP} (resp. G_{VP}) by converting all the hyper edges to the vertices and by converting all the vertices, except for the vertices corresponding to the phantom modules, to the edges. (**Procedure** SeqPair–RDG **End**)

Theorem 5 : Let S be a sequence-pair of n modules. If S does not include adjacent-cross, procedure SeqPair–RDG maps S to a pair of RDGs which correspond to a rectangular-dissection with no empty room such that the HVRS of the rectangular-dissection equals to the HVRS of S, in $O(n^2)$ time. □

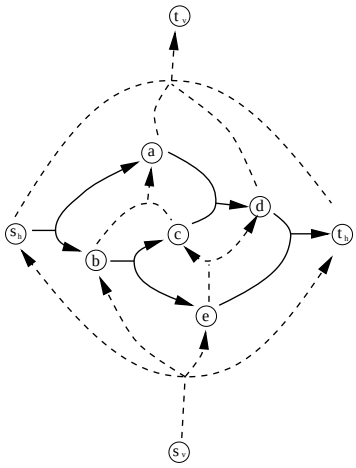
From the resultant RDGs, a rectangular-dissection is obtained by ConstRD in $O(n)$ time. In the following, we prove this theorem.

Lemma 2 : In **(Step 1)**, each edge of G_{HSP} (G_{VSP}) belongs to a unique maximal complete bipartite subgraph of G_{HSP} (G_{VSP}).
Proof : The proof is by contradiction. Assume an edge (a_1, b_1) belongs to two maximal complete bipartite subgraphs $G^1(V_i^1 \cup V_o^1, E^1)$ and $G^2(V_i^2 \cup V_o^2, E^2)$. Since G^1 and G^2 are both maximal complete bipartite graphs, there are two vertices $a_2 \in (V_i^1 \cup V_i^2)$ and $b_2 \in (V_o^1 \cup V_o^2)$ such that there is no edge (a_2, b_2) in $E(G_{HSP})$. The edges (a_1, b_1), (a_1, b_2) and (a_2, b_1) all exist in $E(G_{HSP})$. If a_1 and a_2 are in horizontal relation, then (a_1, b_1) or (a_2, b_1) becomes transitive. Hence a_1 and a_2 are in vertical relation. Without loss of generality, we assume a_1 is above a_2, i.e. $S = (\cdots a_1 \cdots a_2 \cdots, \cdots a_2 \cdots a_1 \cdots)$. Since there are edges (a_1, b_1) and (a_2, b_1), $S = (\cdots a_1 \cdots a_2 \cdots b_1 \cdots, \cdots a_2 \cdots a_1 \cdots b_1 \cdots)$. Considering the fact that there is edge (a_1, b_2), the position of b_2 are exhaustively examined in the following. (See Fig. 3.6).

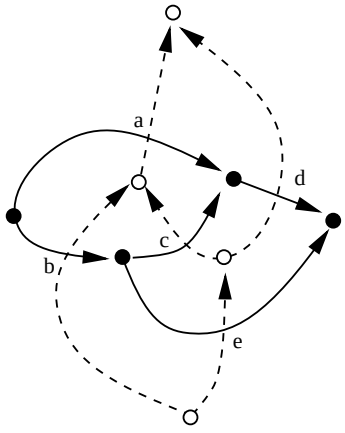
(i) $S = (\cdots a_1 \cdots b_2 \cdots a_2 \cdots b_1 \cdots, \cdots a_2 \cdots a_1 \cdots b_1 \cdots b_2 \cdots)$



(a) G_{HSP} (solid lines) and G_{VSP} (dotted lines) obtained in Step 1



(b) G_H (solid lines) and G_V (dotted lines) obtained in Step 2



(c) G_{HP} (solid lines) and G_{VP} (dotted lines) obtained in Step 3

Figure 3.5: Snapshot of the procedure SeqPair–RDG

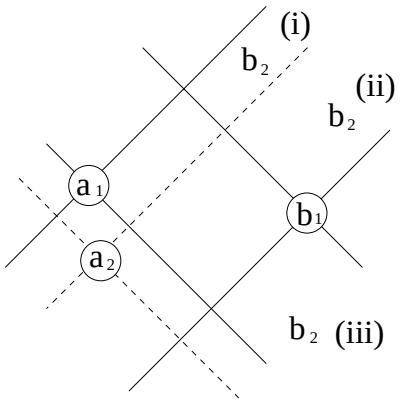


Figure 3.6: Figure used in the proof of Lemma 2

In G_{VSP}, there is a path from a_2 to b_2. An edge in the path crosses to the edge $(a_1, b_1) \in E(G_{HSP})$, thus there is an HV-cross in S. This contradicts to the fact that S has no adjacent-cross (thus no HV-cross by Lemma 1).

(ii) $S = (\cdot\cdot\ a_1 \cdot\cdot a_2 \cdot\cdot b_2 \cdot\cdot b_1, \cdot\cdot\ a_2 \cdot\cdot a_1 \cdot\cdot b_1 \cdot\cdot b_2 \cdot\cdot)$

Since edge (a_2, b_2) does not exist in $E(G_{HSP})$, there is a vertex x which satisfies $S = (\cdot\ a_1 \cdot\ a_2 \cdot\ x \cdot\ b_2 \cdot\ b_1, \cdot\ a_2 \cdot\ x \cdot\ a_1 \cdot\ b_1 \cdot\ b_2)$ or $S = (\cdot\ a_1 \cdot\ a_2 \cdot\ x \cdot\ b_2 \cdot\ b_1, \cdot\ a_2 \cdot\ a_1 \cdot\ b_1 \cdot\ x \cdot\ b_2)$. The former case results in (a_2, b_1) being transitive, and the latter results in (a_1, b_2) being transitive, either contradicts to the definition of G_{HSP}.

(iii) $S = (\cdot\cdot\ a_1 \cdot\cdot a_2 \cdot\cdot b_1 \cdot\cdot b_2 \cdot\cdot, \cdot\cdot\ a_2 \cdot\cdot a_1 \cdot\cdot b_2 \cdot\cdot b_1 \cdot\cdot)$

Since there is no edge (a_2, b_2), there is module x which satisfies $S = (\cdot\cdot\ b_1 \cdot\cdot x \cdot\cdot b_2 \cdot\ \cdot, \cdot\cdot\ a_2 \cdot\cdot x \cdot\cdot a_1)$. Then, there is a path from x to a_1 in G_{VSP}. An edge in the path crosses to the edge $(a_2, b_1) \in E(G_{HSP})$, which is a contradiction.

(iv) The other cases are trivially impossible.

Hence, an edge in G_{HSP} belongs to a unique maximal bipartite subgraph of G_{HSP}. Similarly, the claim also holds for G_{VSP}. □

Lemma 3 : The pair of graphs G_{HP} and G_{VP} obtained by SeqPair–RDG is the RDGs.
Proof : In the following, we show the output is a pair of RDGs by converting the oblique-grid-embedding of $S^\star$. The modules in S are called *real* modules in contrast to the phantom modules.

In the oblique-grid embedding of $S^\star$ which is obtained in **(Step 1)**, any horizontal edge and vertical edge do not cross each other because $S^\star$ does not have HV-cross. (A cross between horizontal edges, or between vertical edges, is possible.)

In the HVRS of $S^\star$, phantom module s_h (t_h, s_v, t_v) is left of (right of, below, above, respectively) every real module. Hence all the vertices corresponding to real modules have at least one input edge and one output edge, both in G_{HSP} and in G_{VSP}.

In the hyper directed graphs G_H and G_V obtained in **(Step 2)**, the input degree and the output degree of each vertex are 0 or 1. From Lemma 2, the input degree and the output degree of the vertices which correspond to real modules are both 1, in either hyper graph. Further, any two edges do not cross each other if they are taken from distinct maximal complete bipartite subgraphs. Hence, G_H and G_V can be drawn without any crossing, as shown in Fig. 3.5-(b).

In **(Step 3)**, the conversion between hyper edges and vertices preserves the planarity, thus G_{HP} and G_{VP} are planar. The input degree and the output degree of G_{HP} and G_{VP} are 1. Hence, the two hyper graphs G_{HP} and G_{VP} are both ordinary graphs. Consequently, G_{HP} and G_{VP} are in polar-dual relation. From Property 2, they are RDGs. □

Lemma 4 : The HVRS of the RDGs obtained by SeqPair–RDG is equivalent to the HVRS of the input sequence-pair.

Proof : In **(Step 1)**, a horizontal (vertical) relation is represented as a path between two vertices in G_{HSP} (G_{VSP}). For each path in G_{HSP} (G_{VSP}), the corresponding path exists in the hyper graph G_H (G_V) in **(Step 2)**, and also in the in G_{HP} (G_{VP}) in **(Step 3)**. No new relation is introduced in the resultant RDGs since the RDGs can not represent both horizontal and vertical relation for a module pair (Property 5). □

(**Proof of Theorem 5**)

Only the speed is proved in the following since the other claims are already proved by Lemma 3 and 4.

In **(Step 1)**, SPGs are constructed faithfully to the HVRS, but eliminating the transitive edges, by its definition. For a module a, the set of all the modules $\{x_1, x_2, \ldots, x_m\}$ that are non-transitively right of module a can be computed in $O(n)$ time using the fact that they are in the form:

$$(\cdot\cdot\ x_m \cdot\cdot x_2 \cdot\cdot x_1 \cdot\cdot, \cdot\cdot\ a \cdot\cdot x_1 \cdot\cdot x_2 \cdot\cdot x_m).$$

Hence, SPGs can be constructed in $O(n^2)$ time.

(Step 2) can be done also in $O(n^2)$ time, proportional to the number of edges in SPGs, because each edge in SPGs belongs to a unique maximal bipartite in SPGs (Lemma 2).

It is obvious that the sum of the cardinality of the input vertex set and that of the output vertex set of all the hyper edges in G_H (G_V) is $O(n)$. Hence, **(Step 3)** can be done in $O(n)$ time.

Consequently, SeqPair–RDG can be done in $O(n^2)$ time. □

3.3.3 Necessary and Sufficient Condition

Theorem 5 shows that the absence of the adjacent-cross is sufficient for a sequence-pair to be mapped to a rectangular-dissection without empty room. It is also necessary as follows.

Theorem 6 : A sequence-pair can be mapped to a rectangular-dissection without introducing any empty room if and only if the sequence-pair does not have an adjacent-cross.

Proof : The condition is sufficient by Theorem 5. Let S be a sequence-pair of n modules and S includes one or more adjacent-crosses. In the following, we show the HVRS of S is not equivalent to the HVRS of any rectangular-dissection with n rooms.

Let four modules a, b, c, d form an adjacent-cross in S. Without loss of generality, Let $S = (\cdot\cdot\, a \cdot\cdot bc \cdot\cdot d \cdot\cdot, \cdot\cdot\, b \cdot\cdot da \cdot\cdot c \cdot\cdot)$. The proof is by contradiction. Assume the relative module position of S is represented by a rectangular-dissection F without any empty room.

In the H-RDG of F, there are three paths;

(i) the path from the edge a to the edge c,

(ii) the path from the edge b to the edge c, and

(iii) the path from the edge b to the edge d.

For the path (ii), from the two facts "b and c are adjacent in Γ_+, and there is no anonymous edge in the H-RDG of F.", it is understood that edge b and edge c are directly connected by a vertex v. It implies that the vertex v is in the path (i) and also in the path (iii). Hence, there is a path from a to d (via v) in the H-RDG. This contradicts to the fact: a and d are in the vertical relation in the HVRS of S, thus not in the horizontal relation. □

3.4 Rectangular-Dissection with Fewest Empty Rooms

In this section, we give a procedure which maps a sequence-pair to a rectangular-dissection with fewest empty rooms. The maximum possible number of empty rooms is also presented.

3.4.1 Removing Adjacent-Crosses

Let $S = (\Gamma_+, \Gamma_-)$ be a sequence-pair of n modules, which possibly includes adjacent-crosses. *Inserting* dummy module x into S is to add a new module x into Γ_+ and into Γ_-. "Adjacent-cross ab/cd" denotes an adjacent-cross such that a and b are adjacent in Γ_+ and c and d are adjacent in Γ_-. For example, $(\cdot\cdot\, d \cdot\cdot ab \cdot\cdot c \cdot\cdot, \cdot\cdot\, a \cdot\cdot cd \cdot\cdot b \cdot\cdot)$ and $(\cdot\cdot\, c \cdot\cdot ab \cdot\cdot d \cdot\cdot, \cdot\cdot\, b \cdot\cdot cd \cdot\cdot a \cdot\cdot)$ are such cases.

For a sequence-pair which includes an adjacent-cross ab/cd, inserting a dummy module x at the *cross-point* of ab/cd indicates that inserting x between a and b in Γ_+ and between c and d in Γ_-. For example, when inserting dummy module x into $(\cdot\cdot\, d \cdot\cdot ab \cdot\cdot c \cdot\cdot, \cdot\cdot\, a \cdot\cdot cd \cdot\cdot b \cdot\cdot)$ at the cross-point of ab/cd, the resultant sequence-pair will be $(\cdot\cdot\, d \cdot\cdot axb \cdot\cdot c \cdot\cdot, \cdot\cdot\, a \cdot\cdot cxd \cdot\cdot b \cdot\cdot)$.

Procedure RmAdjCross

Input: sequence-pair $S = (\Gamma_+, \Gamma_-)$ which possibly has adjacent-crosses.

Output: sequence-pair S which does not have adjacent-cross.

(Step 1) Find an adjacent-cross ab/cd in S. Insert dummy module x at the cross-point of ab/cd. Repeat the above process until no adjacent-cross exists. (**Procedure** RmAdjCross **End**)

Fig. 3.7 illustrates the effect of the procedure. In the figure, white circles indicate the modules in the given sequence-pair, which has three adjacent-crosses whose cross-points

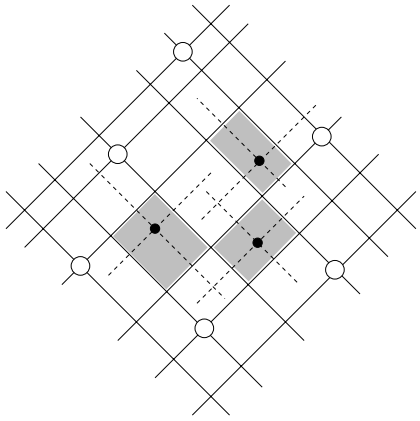


Figure 3.7: Effect of the procedure RmAdjCross. Dummy modules (black dots) are inserted at the cross-points (dark region) of adjacent-crosses.

are remarked by dark color. The black dots indicates the dummy modules inserted by the procedure. It can be examined that the resultant sequence-pair does not have any adjacent-cross.

Using the procedure RmAdjCross, the following theorem is proved in this section.

Theorem 7 : Let S be a sequence-pair. Let F be a rectangular-dissection whose number of rooms is minimum over the rectangular-dissections whose HVRS is the same to the HVRS of S. Such an F can be obtained by RmAdjCross followed by SeqPair–RDG and ConstRD, totally in $O(n^4)$ time. □

Lemma 5 : Let S be a sequence-pair and k be the number of adjacent-crosses in S.

(1) RmAdjCross inserts k dummy modules and the number of adjacent-cross is made zero.

(2) The number of adjacent-cross can not be made zero by $k-1$ or less dummy modules.

Proof :

(1) Suppose a dummy module x is inserted at the cross-point of adjacent-cross ab/cd in S. Let the resultant sequence-pair be S'. Then a, b, c, d do not form an adjacent-cross in S'. (The adjacent-cross is said to be removed.)

Assume a new adjacent-cross is created in S'. One of the four modules which form the new adjacent-cross is x. One of the other three modules is a, b, c or d. Without loss of generality, let a is the one. Let the other two be y and z. Neither of y nor z is a, b, c or d. The new adjacent-cross is then xa/yz. For the adjacent-cross xa/yz in S', there is an adjacent-cross ab/yz in S and it is removed in S'. If there are more new created adjacent-crosses (xa/yz), individual adjacent-crosses are removed (ab/yz). Therefore, the number of adjacent-crosses can be decreased by one by inserting a dummy module at the cross-point of an arbitrary adjacent-cross, which is exactly executed by RmAdjCross.

(2) Suppose there is a sequence-pair $S^\dagger$ which does not include any adjacent-cross but includes only $k-1$ or less dummy modules. If we remove all the dummy modules

from $S^\dagger$, the resultant sequence-pair coincides with S. We remove the dummy modules one by one from $S^\dagger$, and stop when the number of adjacent-crosses is increased by two or more by removing the dummy module x. Then, if we insert x exactly at the position it has been existed, the number of adjacent-crosses should be decreased by two or more. We show this can not be happened, in the following.

Let a dummy module x be inserted to S and m adjacent-crosses be removed. When an adjacent-cross ab/cd is removed, (i) x is inserted between a and b in Γ_+, or (ii) x is inserted between c and d in Γ_-. Both of the conditions are true at most for one adjacent-cross. Thus at least $m-1$ adjacent-crosses satisfy either (i) or (ii). Let adjacent-cross ab/cd be one of those adjacent-crosses. Then, x and three modules from a, b, c, d form a new adjacent-cross in S' (such as xb/cd). This new created adjacent-cross (xb/cd) exists individually for all $m-1$ adjacent-crosses. Thus, the number of adjacent-crosses can be decreased at most by one by inserting one dummy module. □

Lemma 6 : Let S be a sequence-pair. Let F be a rectangular-dissection whose number of rooms is minimum over the rectangular-dissections whose HVRS is the same to the HVRS of S. Such an F can be obtained by RmAdjCross, followed by SeqPair–RDG and ConstRD

Proof : Since the sequence-pair obtained by RmAdjCross does not include adjacent-cross, a rectangular-dissection F' is obtained by SeqPair–RDG and ConstRD. In the following, we show the number of rooms in F' equals to that of F. From Property 6, any rectangular-dissection with n modules, possibly has empty rooms, corresponds to unique sequence-pair of n module names, preserving the HVRS. Then if we assign dummy modules to all the empty rooms in F, we have a unique sequence-pair S' with modules corresponding to all the rooms including the empty rooms. From Theorem 6, S' does not have an adjacent-cross. The HVRSs of S and S' (with respect to the pre-existing modules) are the same because they are the same to the HVRS of F. Thus if we remove all the dummy modules from S', it coincides with S. Since the number of dummy modules inserted by RmAdjCross is minimum to remove all the dummy modules (Lemma 6), the number of rooms in F and that of F' are the same. □

(**Proof of Theorem 7**)

Only the time complexity is proved in the following since the other claims are already proved by Lemma 6.

It is separately shown in the next section that the maximum number of adjacent-crosses is $O(n^2)$. For each adjacent-cross, RmAdjCross can identify the adjacent-cross in $O(n^2)$ time, and insert a dummy module in $O(n)$ time. Thus, RmAdjCross can be done in $O(n^4)$ time. Since the number of modules in the resultant sequence-pair is $O(n^2)$, SeqPair–RDG runs in $O(n^4)$ time, and ConstRD runs in $O(n^2)$ time. □

The complexity of RmAdjCross can be reduced to $O(n^2)$ if the adjacent-crosses are removed in a certain order. However, the overall complexity is not reduced because SeqPair–RDG dominates the total complexity.

3.4.2 Maximum Number of Empty Rooms

Theorem 8 : Let S be a sequence-pair of n modules. Let F be a rectangular-dissection whose number of rooms is minimum over the rectangular-dissections whose HVRS are the same to the HVRS of S. The maximum possible number of empty rooms in F is

$$\left\lceil \frac{n-2}{2} \right\rceil \left\lfloor \frac{n-2}{2} \right\rfloor .$$

Proof : The proposition is true for $n \leq 3$. (No empty rooms are needed.) We assume $n \geq 4$ in the following.

Without loss of generality, we assume $\Gamma_+ = (1, 2, 3, \ldots, n)$ and $\Gamma_- = (a_1, a_2, a_3, \ldots, a_n)$. A necessary condition to form an adjacent-cross ab/cd is, a and b are adjacent in Γ_+, $c < \min(a, b)$, $d > \max(a, b)$, and $\Gamma_-^{-1}(c) < \Gamma_-^{-1}(d)$ if $b < a$, $\Gamma_-^{-1}(c) > \Gamma_-^{-1}(d)$ otherwise.

Thus, the number of empty rooms can not exceed

$$\sum_{i=2}^{n} \min(i-2, n-i) = \left\lceil \frac{n-2}{2} \right\rceil \left\lfloor \frac{n-2}{2} \right\rfloor .$$

Given n, the sequence-pair constructed as follows has exactly $\lceil (n-2)/2 \rceil \lfloor (n-2)/2 \rfloor$ adjacent-crosses. Γ_+ is constructed as $(1, 2, 3, \cdot, n)$. If n is even, then Γ_- is constructed as

$$\Gamma_-(i) = \begin{cases} \frac{n+1}{2} + (-1)^i (i - \frac{1}{2}) & \text{if } i \leq \frac{n}{2} \\ \frac{n+1}{2} + (-1)^i (n + \frac{1}{2} - i) & \text{otherwise} \end{cases} .$$

If $n = 4k + 1$ for some k, then

$$\Gamma_-(i) = \begin{cases} \frac{n}{2} + (-1)^i (i - \frac{1}{2}) & \text{if } i \leq \frac{n-1}{2} \\ \frac{n}{2} + 1 + (-1)^i (n + \frac{1}{2} - i) & \text{otherwise} \end{cases} .$$

If $n = 4k + 3$ for some k, then

$$\Gamma_-(i) = \begin{cases} \frac{n}{2} + (-1)^i (i - \frac{1}{2}) & \text{if } i \leq \frac{n+1}{2} \\ \frac{n}{2} + 1 + (-1)^i (n + \frac{1}{2} - i) & \text{otherwise} \end{cases} .$$

It is easily examined that the resultant sequence-pair has $\lceil (n-2)/2 \rceil \lfloor (n-2)/2 \rfloor$ adjacent-crosses. □

For example, for $n = 10$, the above construction results in:

$$(\Gamma_+, \Gamma_-) = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10,\ 5\ 7\ 3\ 9\ 1\ 10\ 2\ 8\ 4\ 6).$$

Fig. 3.8 shows the corresponding oblique-grid-embedding of the sequence-pair and the corresponding rectangular-dissection with 16 empty rooms.

3.5 Conclusion

In Chapter 2, sequence-pair is introduced as a data structure to represent candidate solutions for a placement problem. In spite of its efficiency in representing the modules positions, no information is provided for channel positions. Such a channel information

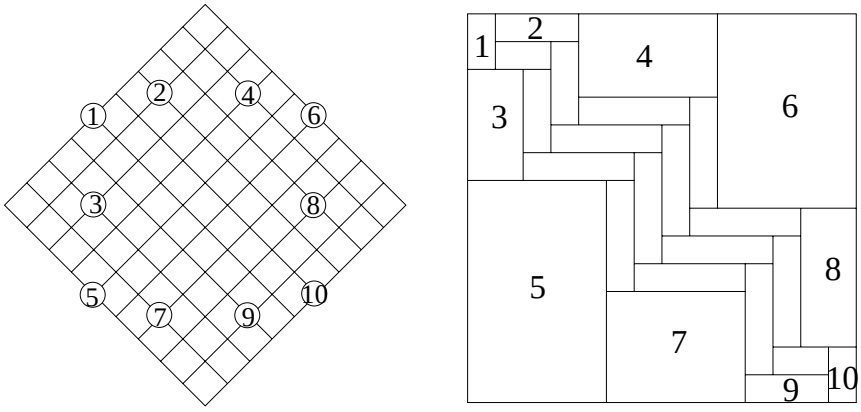


Figure 3.8: Sequence-pair with maximum adjacent-crosses (left) and its corresponding rectangular-dissection (right)

is added in this chapter by introducing a mapping from a sequence-pair to a rectangular-dissection, which has been used to represent the channel positions together with the module positions.

The result of this chapter is summarized as follows.

- Channels are additionally represented, without changing the information about module positions, thus the following two properties of the sequence-pair are remain effective; an area minimum placement is represented, and no overlapping placement is represented.
- The number of channels is minimized, which most likely minimizes the number of wire bends, later in the routing stage.
- The maximum possible number of empty rooms, which linearly corresponds to the maximum possible number of introduced channels, is presented.
- A necessary and sufficient condition of the sequence-pair for not introducing any empty-room is presented.

Although the channels are represented, this chapter does not give a technique to assign width to each channel. How to assign adequate widths to the channels remains hard, and would be solved heuristically.

Chapter 4

Rectangle Packing with Obstacles

4.1 Introduction

In VLSI design, it often happens that the locations of some macro cells, such as RAM, ROM, and CPU core, are fixed a priori and the other components are subject to be placed in the rest of the chip area. Also in PCB design, it is common that the exact coordinates of connectors are determined before designing the placement of the other components. We formulate such situations as a problem called "rectangle packing with pre-placed rectangles (RPP)". Not only the circuit components but also rectangular obstacles in any type are candidates to be modeled as pre-placed modules. For example, pre-placed modules can be used for representing a rectilinear substrate and holes of the substrate. The other "free" modules are requested to be placed onto the substrate without any overlap with the pre-placed modules.

Chi [14] studied a similar but restricted problem, where all the free modules have a regular height, assuming they are standard cells. Force-Directed-Relaxation method (FDR) [6, 7] can be easily tailored to handle the obstacles, but the method has inherent defects 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 [8] is not useful for **RPP**, since the pre-placed modules might be given non-slicibly. For general (including both slicible and non-slicible) placements, we proposed the coding coding schemes called sequence-pair in Chapter 2. Using the sequence-pair, it is easy to generate non-overlapping placements of all the modules by encoding all the (free and pre-placed) modules, but the code could be inconsistent to recover the locations of the pre-placed modules.

In this paper, we present a procedure called adaptation, which changes a sequence-pair so that it becomes consistent to the pre-placed modules. The procedure only changes the positions of pre-placed modules in a sequence-pair, and it runs in $O(n^2)$ time, where n is the total number of pre-placed modules and free modules. The alternative ways to incorporate the adaptation procedure in a simulated annealing are demonstrated through experiments on an MCNC building block benchmark example, named *ami49*. The simulated annealing is applied 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 4.2 gives a formal definition of **RPP**, and addresses that a sequence-pair can be inconsistent to **RPP**. In Section 4.3, the

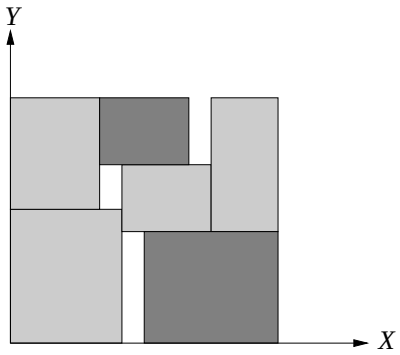


Figure 4.1: Feasible packing example. The dark rectangles are the pre-placed modules.

adaptation procedure is presented. Section 4.4 is devoted for the experiments. Section 4.5 is for conclusion.

4.2 Preliminary

4.2.1 Rectangle Packing with Pre-Placed Rectangles (RPP)

A *module* is a rectangle. A *packing* of a set of modules is a non-overlapping placement of the modules. The coordinates of a module is the coordinates of the lower left corner of the module. A *pre-placed* 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 coordinates is not specified. Set P of pre-placed modules is given such that no two pre-placed 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. Find the best feasible packing of $P \cup F$.

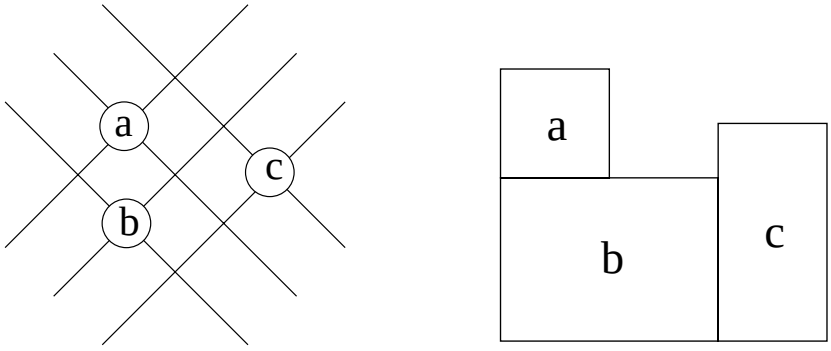
Fig. 4.1 shows a feasible packing of an instance of **RPP**.

When all the modules are free, i.e. $P = \emptyset$, then **RPP** coincides with the packing problem discussed in Chapter 2, denoted by **RP**, which is already proved to be $\mathcal{NP}$-hard. Thus, **RPP** is also in $\mathcal{NP}$-hard class.

4.2.2 Sequence-Pair

For the free packing problem (**RP**), a coding scheme is proposed in Chapter 2 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 (abc, bac) (b) free-realization

Figure 4.2: Oblique grid notation of a sequence-pair and the free-realization of the sequence-pair

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

$$\begin{aligned}(\cdot\cdot\, a \cdot\cdot b \cdot\cdot, \cdot\cdot\, a \cdot\cdot b \cdot\cdot) &\rightarrow a \text{ should be placed to the left of } b\\ (\cdot\cdot\, b \cdot\cdot a \cdot\cdot, \cdot\cdot\, a \cdot\cdot b \cdot\cdot) &\rightarrow a \text{ should be placed below } b\end{aligned}$$

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. 4.2(a) shows the oblique-grid of sequence-pair (abc, bac). It is an $n \times n$ grid obliquely drawn on the plane, so constructed 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 in such a way that: Modules c is in the right quarter view range (between -45 degree and $+45$ degree) of module a, then c should be placed to the right of a.

It is proved in Chapter 2 that: The set of H/V constraints imposed by every sequence-pair is satisfiable, and 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. 4.2(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)

[1] The time complexity is reduced to $O(n \log n)$ by Takahashi[22].

Construct 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 widths of the corresponding module. (Step 2) Determine X coordinate of each module by the longest path length from the source nodes to the node of the module. (Step 3 and Step 4) Determine Y coordinate of each module in a similar way using the vertical constraints in the direction from bottom to top.

4.2.3 Feasibility of Sequence-Pair

Now we assume $P \neq \emptyset$, i.e., one or more modules are pre-placed. If we only encode the free modules into a sequence-pair, a free module and a pre-placed module can easily overlap each other in the free-realization of the sequence-pair, since the pre-placed module is totally ignored. Fig. 4.3(a) illustrates such an example for this situation. In the figure, free modules a, b, c, d are placed without considering the pre-placed 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 pre-placed modules into a sequence-pair. One difficulty in the free-realization of such a sequence-pair is, the X (as well as Y) coordinate of a pre-placed module may be set too small, because modules are compacted to left (bottom) without considering the pre-assigned coordinates. Fig. 4.3(b) shows such an example, where the X and Y coordinates of pre-placed module x are set too small, and the X coordinate of pre-placed module y is set too small. This difficulty is easily solved by introducing an additional constraint for each pre-placed module:

> The X (Y) coordinate of each pre-placed module should not be less than the specified value.

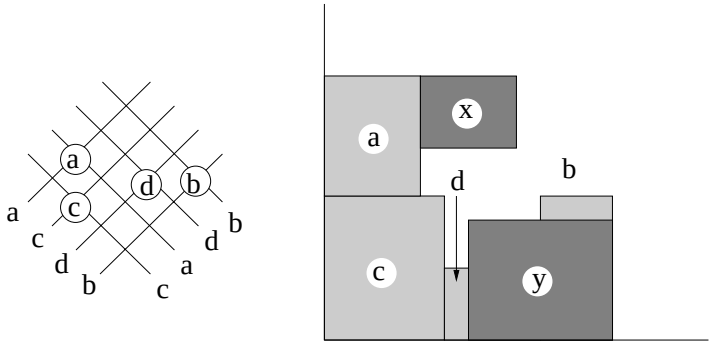
The free-realization procedure is easily modified to handle the additional constraints without increasing the asymptotic speed, by adding additional edges from source to the pre-placed modules, with the pre-assigned coordinates as their weights, in the H/V constraint graph (Fig. 4.3(c)). The resultant placement is now called the *propped-realization*, named from an intuitive image of the additional constraints.

Fig. 4.3(c) shows the propped-realization for the same example in Fig. 4.3(b). The arcs in the oblique-grid denote the additional constraints. In the figure, pre-placed module x is placed at the specified location. However, pre-placed module y is not placed at the specified location (X coordinate is set too large). This is because the additional constraints introduced above can not prevent the coordinates being set too large. It is impossible to reform the X coordinate of module y by adding another constraint, since the sequence-pair inherently imposes y should be placed right of x. In the following, a sequence-pair is said *feasible* when its propped-realization is feasible, otherwise *infeasible*.

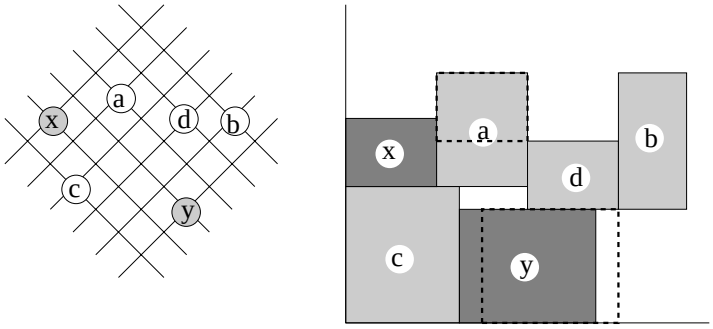
It is concluded that a sequence-pair is not necessarily feasible for **RPP**. However, it is still useful from the following fact.

Lemma 7 : For any instance of **RPP**, there is a feasible sequence-pair. Furthermore, there is a sequence-pair which leads an optimal solution of the problem.

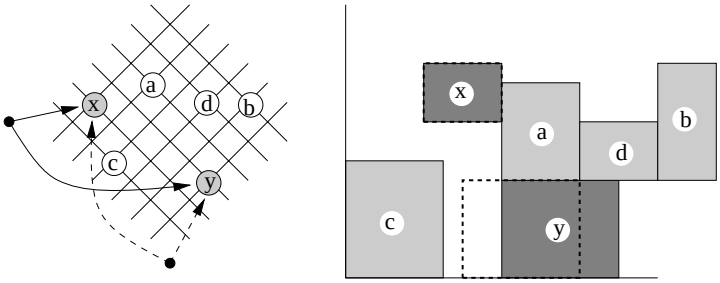
Proof : A proof for the second claim suffices as a proof for the first claim. Let Π be an optimal solution for **RPP** with pre-placed modules P and free modules F. From



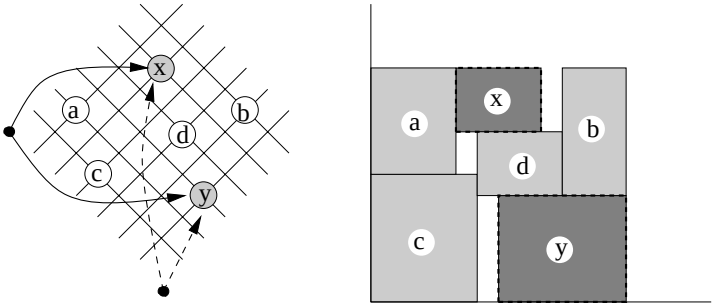
(a) sequence-pair of free modules



(b) sequence-pair of free and pre-placed modules



(c) additional constraints for pre-placed modules



(d) feasible sequence-pair

Figure 4.3: Feasibility of Sequence-Pair

Chapter 2, 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 difference between X/Y coordinates of the modules in Π, if the path consists of H/V constraint edges only. Since the weight of each additional edge equals to the X (Y) coordinate of the pre-placed 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. 4.3(d) shows an example of a feasible (and optimal) sequence-pair and its propped-realization.

4.3 Adaptation

Among $(n!)^2$ sequence-pairs, there are feasible sequence-pairs, including one for optimal solution of **RPP**, and infeasible sequence-pairs. We should consider how to treat the infeasible sequence-pairs, when exploring the space by a stochastic algorithm. The easiest 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 to 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 pre-placed modules.

4.3.1 Necessary Condition

Suppose unit square modules a and b are pre-placed at $(3, 3)$ and at $(5, 5)$, respectively. Then, both of "a is left of b" and "a is below b" are true, but "a is right of b" and "a is above b" are both false. It turns out that a is necessary to appear before b in the second sequence of a feasible sequence-pair. More in general, we describe a necessary condition for the second sequence of a feasible sequence-pair.

For any two pre-placed modules a and b, we say that a *dominates* b if

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

where

- $x(a), y(a)$: X and Y coordinates of module a,
- $w(a), h(a)$: width and height of module a.

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

Lemma 8 : It is necessary for a sequence-pair being feasible that the second sequence is topologically sorted. □

4.3.2 Algorithm

In the propped-realization of a sequence-pair, the X and Y coordinates of a module is 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 "test and virtually place" a module according to the second sequence. Thus, after some iterations, free modules would be virtually placed, as well as pre-placed modules. The relation "dominate" was already defined for the pre-placed modules, but in the following, we use the relation also for a free module when it is virtually placed.

Procedure Adaptation

Input: A set of pre-placed modules, a set of free modules, a sequence-pair.

Output: A feasible sequence-pair.

(Step 1) Topologically sort the second sequence, only when it is not already topologically sorted.

(Step 2) For $k = 1, 2, \ldots, n$, repeat **(Step 2.1)** through **(Step 2.4)**.

(Step 2.1) Let the k'th module in the second sequence be a. If module a is a pre-placed module, go to **(Step 2.4)**. Otherwise, temporary set the coordinates of a, according to the propped-realization of a.

(Step 2.2) Determine whether there exists a pre-placed module which dominates a, in the last $n - k$ modules of the second sequence. When it exists, determine a pre-placed module q such that q directly or transitively dominates a, and there is no pre-placed module which dominates q. Otherwise, go to **(Step 2)** for the next iteration.

(Step 2.3) In the second sequence, move q just before a. After that, a is used for referring q. (since q is now the k'th module in the second sequence.)

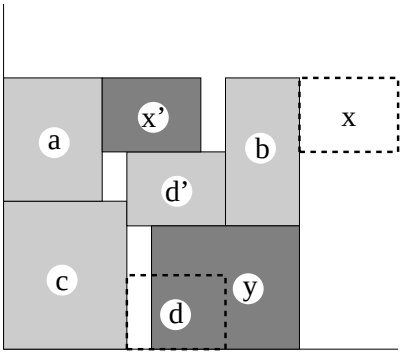
(Step 2.4) If a is a free module, go to **(Step 2)** for the next iteration. Let (x, y) be the coordinates of a in the propped-realization. Let $(x(a), y(a))$ be the specified coordinates of pre-placed module a. If $x > x(a)$, then in the first sequence, move a minimally toward the top so that $x \leq x(a)$ holds. Otherwise, if $y > y(a)$, then in the first sequence, move a minimally toward the end so that $y \leq y(a)$ holds.

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

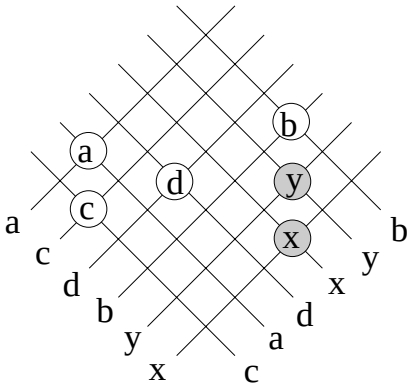
4.3.3 Illustrative Example

A behavioral example is presented for the **RPP** instance shown in Fig. 4.4(a), in which the dark two modules x and y are pre-placed, and the rest of the modules a, b, c, d are free.

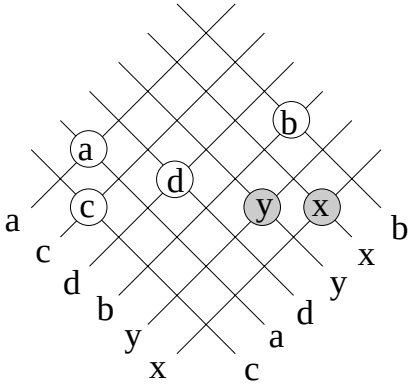
Suppose sequence-pair $(acdbyx, cadxyb)$ is given as the input sequence-pair. Fig. 4.4(b) shows this initial sequence-pair. In **(Step 1)**, x and y are exchanged in the second sequence, and the sequence-pair becomes $(acdbyx, cadyxb)$, as shown in Fig. 4.4(c).



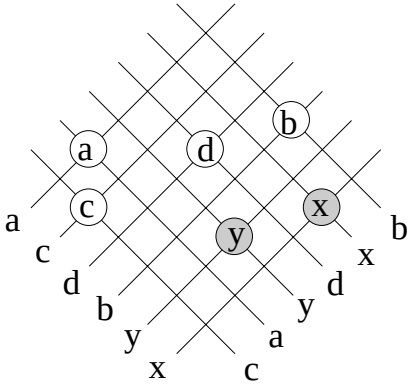
(a) The packing example



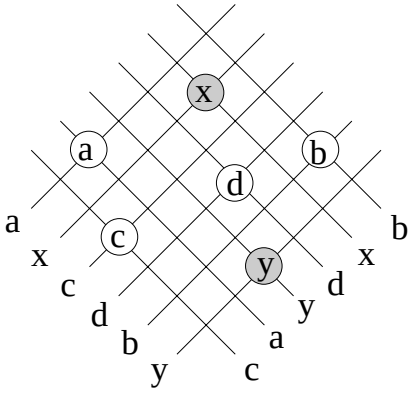
(b) $(acdbyx, cadxyb)$



(c) $(acdbyx, cadyxb)$



(d) $(acdbyx, caydxb)$



(e) $(axcdby, caydxb)$

Figure 4.4: Snapshots of the adaptation procedure. (a) shows the packing corresponding to the sequence-pair shown in (e). (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.

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. 4.4(a).

In iteration $k = 3$, module d is once tried to be placed at the right of c in **(Step 2.1)**, as indicated by the rectangle with dotted lines, which is marked "d" in Fig. 4.4(a). Then, module y is selected in **(Step 2.2)**, and it is brought before d in the second sequence in **(Step 2.3)**. As a result, the sequence-pair becomes $(acdbyx, caydxb)$ as shown in Fig. 4.4(d). Module y is virtually placed at its specified position, as it is marked "y" in Fig. 4.4(a).

In iteration $k = 4$, module d is placed at the position marked "d'"in Fig. 4.4(a).

In iteration $k = 5$, pre-placed module x is processed. In **(Step 2.4)**, it is once tried to placed at the position indicated by the rectangle with dotted lines, which is marked "x" in Fig. 4.4(a), since it is the position imposed by the sequence-pair shown in Fig. 4.4(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. 4.4(e). Module x is virtually placed at its specified position, as it is marked "x'" in Fig. 4.4(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.

4.3.4 Proof of Adaptation

Theorem 9 : 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 pre-placed modules and the free modules.

Proof :

(1) and (3) are easily understood. (2) and (4) are proved in the following.

(2): Adaptation outputs a feasible sequence-pair.

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 $M - M^k$.

When the k'th iteration starts in **(Step 2)**, assume that the second sequence is topologically sorted with respect to the domination relation between pre-placed 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 2.4)** determines the k'th 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 2.4)** produces a feasible S^k (for M^k) in k'th iteration.

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

Module a should be a pre-placed 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 2.4)** is successfully executed. So, there must 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 pre-placed 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 2.3)**, 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 2.2)**, hence b is not dominated by any other module. This contradicts to 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 pre-placed module in **(Step 2.2)**. This contradicts to 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, essentially because of the fact: If a transitively dominates b and $x(a) < x(b)$, 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 proved also by the ordinary depth first algorithm, but we use the above algorithm with utmost consideration for minimizing the modification.)

(Step 2.1) is easily done in $O(n)$ time. **(Step 2.2)** can be done in $O(n)$ time by traversing the modules reversely in the second sequence. **(Step 2.3)** can be easily done in $O(n)$ time. **(Step 2.4)** can be done in $O(n)$ time by introducing two arrays $X[1 \ldots n]$ and $Y[1 \ldots n]$ which holds a "negative locus" (see Chapter 2) of the k'th module in the second sequence after the module is virtually placed.

(Step 2.1) through **(Step 2.4)** is repeated n times in **(Step 2)**. Therefore, the adaptation can be done in $O(n^2)$ time. □

4.4 Use of Adaptation

4.4.1 RPP Example

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 pre-placed 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, and
- 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.
- 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.

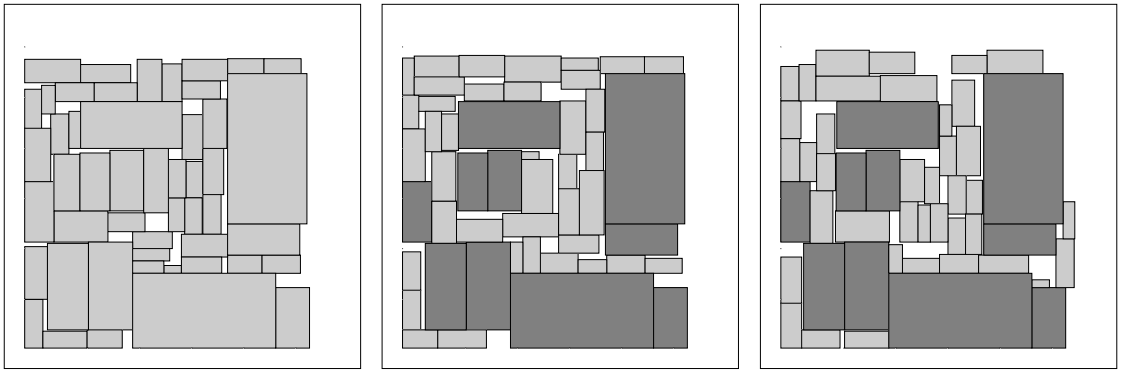


Figure 4.5: Three packings of ami49. The left figure is obtained by RP. The middle and the right figures are obtained by Adapt-in-Eval method and Adapt-in-Move method, respectively, setting the dark modules being pre-placed.

Table 4.1: 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

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.

The biggest MCNC building block benchmark example *ami49* is used as the input data. As a preliminary run, the 49 modules are all treated as free modules, and they are packed by the simulated annealing. The first 10 biggest modules, with respect to the areas, are specified as pre-placed modules whose coordinates are fixed to those obtained by the preliminary run. The reason why these modules are selected is because of our experience that the obstacles are usually not many, but big. The same input, except for the presence of the pre-placed modules, is applied to Adapt-in-Eval method and to Adapt-in-Move method.

Fig. 4.5 shows the resultant packing of these methods, including the preliminary run. Table 4.1 shows the average performance of 10 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 is shortest in **RP**, as in Table 4.1. 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 pre-placed modules, with an algorithm similar to [22]. The reason why Adapt-in-Move runs faster than Adapt-in-Eval method would be because the input sequence-pair

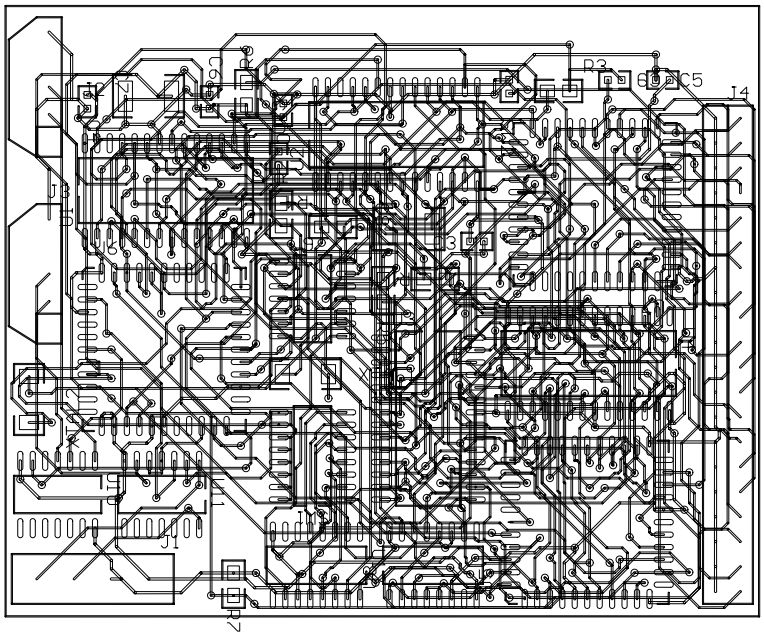


Figure 4.6: A place-and-route result of DEMOSMD. Connectors J1,J2,J3 and J4 are pre-placed.

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. This implies that the adaptation can not compensate completely the inconsistency introduced by the pre-placed modules. It is also observed from Table 4.1 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.

4.4.2 Place and Route Example

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

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, with weighting coefficients which are assigned so that the term of the area and the term of the estimated wiring length are approximately balanced.

A PCB example which includes 36 modules and 88 multi-terminal 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

Table 4.2: 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

data to try $100,000$ moves, on a note-type personal computer (Sharp Mebius PC-A355, Pentium 100MHz). The resultant placement was given to a commercial router (Protel AdvancedRoute3) to finish the design using 4 layers.

All the nets were successfully routed, as shown in Fig. 4.6. The performance is summarized in Table 4.2, in the column labeled with "MST".

By replacing the MST estimation with the half perimeter of the minimum bounding box, which is a popular way to speed up the estimation, the same trial was carried out and also completely routed by the router. Results are in Table 4.2, in the "MBOX" column.

4.5 Conclusion

We showed that the presence of the pre-placed modules introduces an inconsistency to the sequence-pair coding scheme. An adaptation procedure is proposed to transform an inconsistent sequence-pair to a consistent one. It is shown that a simulated annealing is well organized to test only feasible placements with this adaptation procedure. A PCB example is placed with a standard wiring estimation, and is routed successfully by a commercial router.

Chapter 5

Conclusion

The motivation for this work was our experience that many VLSI designers are not satisfied with the slicing structure and that many PCB designers place the modules manually. To break-though this difficulty, a fundamental rectangle packing problem is investigated in this thesis.

The major contribution of this thesis is to present a representation of a general packing in terms of a pair of module name sequences, called sequence-pair. Detailed proofs are presented to show that every sequence-pair feasibly corresponds to a packing, and at least one sequence-pair corresponds to an area-minimum packing.

In experiments, 500 rectangles were packed very efficiently in a reasonable time. It was attained by a standard simulated annealing in which a move is a change of the sequence-pair. The evaluating function was then extended to the VLSI placement problem using a conventional wiring area estimation method. It shows such a performance that the biggest MCNC benchmark, ami49, is placed promisingly.

Another effort is devoted to fill in the gap between the fundamental packing problem and the practical placement problems. In practice, there are various additional criteria and constraints besides the basic criterion of packing (chip area). This thesis gives methods for two of those topics, channel generation and placement with obstacles.

New VLSI technologies are offering a sufficient number of layers for routing to allow enough space for routing to be done on top of the modules, assuming the use of some "area router". This technology trend will bring placement design still closer to packing, hence make our approach more significant.

However, practical layout in VLSI/PCB is processed under various constraints and different objectives. For example, wire congestion estimation, timing, power dissipation, clock distribution, crosstalk, are crucial to consider. To include them in our pure packing algorithm, many studies on real instances and theories are necessary, which are left for future researches.

Bibliography

[1] B. S. Baker, E. G. Coffman, and R. L. Rivest, "Orthogonal Packings in Two Dimensions," *SIAM J. Comput.*, vol. 9, no. 4, pp. 846–855, 1980.

[2] I. Golan, "Performance Bounds for Orthogonal Oriented Two-Dimensional Packing Algorithms," *SIAM J. Comput.*, vol. 10, no. 3, 1981.

[3] B. S. Baker, D. J. Brown, and H. P. Katseff, "A 5/4 Algorithm for Two-Dimensional Packing," *Journal of Algorithms*, vol. 2, pp. 348–368, 1981.

[4] E. G. Coffman and J. C. Lagarias, "Algorithms for Packing Squares: A Probabilistic Analysis," *SIAM J. Comput.*, vol. 18, no. 1, pp. 166–185, 1989.

[5] I. Schiermeyer, "Reverse-Fit: A 2-optimal Algorithm for Packing Rectangles," *ESA'94 Algorithms: Lecture Notes in Computer Science*, vol. 855, pp. 290–299, 1994.

[6] L. Sha and R. W. Dutton, "An Analytical Algorithm for Placement of Arbitrarily Sized Rectangular Blocks," in *Proc. 22th ACM/IEEE Design Automation Conf.*, pp. 602–608, 1985.

[7] A. Alon and U. Ascher, "Model and Solution Strategy for Placement of Rectangular Blocks in the Euclidean Plane," *IEEE Trans. on CAD*, vol. 7, no. 3, pp. 378–386, 1988.

[8] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Designs," in *Proc. 23rd ACM/IEEE Design Automation Conf.*, pp. 101–107, 1986.

[9] H. Esbensen and E. S. Kuh, "An MCM/IC Timing-Driven Placement Algorithm Featuring Explicit Design Space Exploration," in *Proc. of the IEEE Multi-Chip Module Conference*, pp. 170–175, 1996.

[10] R. H. J. M. Otten, "Automatic Floorplan Design," in *Proc. 19th ACM/IEEE Design Automation Conf.*, pp. 261–267, 1982.

[11] T. C. Wang and D. F. Wong, "An Optimal Algorithm for Floorplan Area Optimization," in *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 180–186, 1990.

[12] W. M. Dai and E. Kuh, "Simultaneous Floorplanning and Global Routing for Hierarchical Building Block Layout," *IEEE Trans. on CAD*, vol. CAD-6, no. 5, pp. 828–837, 1987.

[13] H. Onodera, Y. Taniguchi, and K. Tamaru, "Branch-and-Bound Placement for Building Block Layout," in *Proc. 28th ACM/IEEE Design Automation Conf.*, pp. 433–439, 1991.

[14] M. C. Chi, "An Automatic Rectilinear Partitioning Procedure for Standard Cells," in *24th ACM/IEEE Design Automation Conference*, pp. 50–55, 1987.

[15] Y. Kajitani, "—," in *personal communication*, 1997.

[16] Y. Kajitani, "Order of Channels for Safe Routing and Optimal Compaction of Routing Area," *IEEE Trans. on CAD*, vol. 2, no. 4, pp. 293–300, 1983.

[17] L. Stockmeyer, "Optimal Orientations of Cells in Slicing Floorplan Designs," *Information and Control*, vol. 59, pp. 91–101, 1983.

[18] P. Pan, W. Shi, and C. L. Liu, "Area Minimization for Hierarchical Floorplans," in *IEEE International Conf. on Computer Aided Design*, pp. 436–440, 1994.

[19] M. J. Ciesielski and E. Kinnen, "Digraph Relaxation for 2-Dimensional Placement of IC Blocks," *IEEE Trans. on CAD*, vol. CAD-6, no. 1, pp. 55–66, 1987.

[20] S. M.Sait and H. Youssef, *VLSI Physical Design Automation*. IEEE Press, 1995.

[21] S. Wimer, I. Koren, and I. Cederbaum, "Floorplans, Planar Graphs, and Layouts," *IEEE Trans. Circuits & Syst.*, vol. 35, no. 3, pp. 267–278, 1988.

[22] T. Takahashi, "An Algorithm for Finding a Maximum-Weight Decreasing Sequence in a Permutation, Motivated by Rectangle Packing Problem," *Technical Report of IEICE*, vol. VLD96, no. 201, pp. 31–35, 1996.

Publications

[1] S. Nakatake, H. Murata, K. Fujiyoshi, and Y. Kajitani, "Bounded-Slicing Structure for Module Placement", *Technical Report of IEICE*, Vol. VLD94, No. 313, pp.19–24, 1994.

[2] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "A Solution Space of Size $(n!)^2$ for Optimal Rectangle Packings", *8th IEICE Circuit and Systems Karuizawa Workshop*, pp.109–114, 1995.

[3] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-Packing-Based Module Placement", *1995 IEEE/ACM International Conference on Computer-Aided Design*, pp.472–479, 1995.

[4] T. Watanabe, H. Murata, and K. Fujiyoshi, "An Optimal Floorplan with Module Relations Represented by a Sequence-Pair", *9th IEICE Circuit and Systems Karuizawa Workshop*, pp.103–108, 1996.

[5] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module Placement on BSG-Structure and IC Layout Applications", *1996 IEEE/ACM International Conference on Computer-Aided Design*, pp. 484–491, 1996.

[6] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement Based on Rectangular-Packing by the Sequence-Pair", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 12, pp.1518–1524, Dec. 1996.

[7] H. Murata, K. Fujiyoshi, T. Watanabe, and Y. Kajitani, "A Mapping from Sequence-Pair to Rectangular Dissection", *Asia and South Pacific Design Automation Conference*, pp. 625–633, Jan. 1997.

[8] H. Murata, K. Fujiyoshi, and M. Kaneko, "VLSI/PCB Module Placement with Obstacles Based on Sequence-Pair", *1997 International Symposium on Physical Design*, Apr. 1997. (to appear)