HIERARCHICAL FLOORPLANNING FOR VLSI BUILDING BLOCK LAYOUT

Michiroh Ohmura

April 1991

Dissertation submitted to the Graduate School of Engineering of Hiroshima University in partial fulfillment of the requirements for the degree of Doctor of Engineering.

ABSTRACT

Over the past three decades, the remarkable growth of the semiconductor industry has been continued, which started from the invention of the transistor, and now realizes an industry that is all-pervasive in its effect on modern life. Moore's law shows a resultant doubling of circuit complexity each year. With the increase of the demand to integrate the whole system into one VLSI chip, Application Specific Integrated Circuit (ASIC) has emerged since early 1980s. In addition to the reduction of the design period and the cost, designing the high performance ASIC with millions of gate circuits will require a new Computer Aided Design (CAD) method.

This dissertation discusses hierarchical floorplanning for VLSI building block layout, and proposes a new hierarchical floorplanning method combined with global routing and positioning.

In Chapter 1, the outline of the VLSI layout design is discussed as a background of hierarchical floorplanning. In particular, placement and floorplanning in the VLSI layout design are explained in detail.

In Chapter 2, hierarchical floorplanning is explained in detail. First, conventional hierarchical floorplanning methods are introduced. Then difficulties of these floorplanning methods are pointed out. Finally, a new hierarchical floorplanning method combined with global routing and positioning is proposed, and the outline of this method is mentioned.

In Chapter 3, some theoretical results concerning the proposed hierarchical floorplanning method are discussed. The first result is an initial placement method by the ideal distance, which is a newly introduced concept. The second result is overlap resolution in which relative positions of modules are preserved. This is formulated as an overlap resolution problem and proved to be $NP - complete$. A heuristic algorithm is also given. The third result is improvement of one dimensional module placement. This is formulated as the improvement problem and for this problem, an optimum algorithm and its proof are given.

The proposed hierarchical floorplanning consists of the initial floorplanning and the detailed floorplanning. In Chapter 4, algorithms of each phase in the initial floorplanning and the detailed floorplanning are explained with some experimental results, which verify the effectiveness of these algorithms.

In Chapter 5, a prototype system FLORA II, which is constructed based on the proposed hierarchical floorplanning method, is introduced, and some experimental results concerning the hierarchical floorplanning system FLORA II are given. Not only floorplans but also final layouts after global routing is compared with a conventional method.

Finally, the conclusions of this dissertation and the future research works are discussed in Chapter 6.

ACKNOWLEDGMENTS

During the course of this work, I have been fortunate to have received assistance from many individuals. I especially would like to thank my supervisor Professor Noriyoshi Yoshida for his continuing support, encouragement and guidance for this work. I would also like to thank Associate Professor Shin'ichi Wakabayashi, Research Associate Dr. Jun'ichi Miyao of Hiroshima University, and Professor Tohru Kikuno of Osaka University for their invaluable coordination and support throughout the research work for this dissertation.

I am very grateful to the member of my dissertation reviewing committee: Professor Tadashi Ae, Professor Atsuo Fujimoto, Professor Kenji Onaga, and Professor Hiroshi Sasaki, for their careful reading and invaluable comments of this dissertation.

I also thank other faculty members, Professors Tadao Ichikawa, Mitsuo Nagamachi, Eihachiro Nakamae, Akira Nakamura, Mitsuo Ohta, Shunji Osaki, Masatoshi Sakawa, and Noriaki Setō for their guidance and lectures throughout my graduate school career.

I have also received many insightful comments and valuable discussions for main topics of this dissertation through international conferences and private communication. In particular, I would like to acknowledge Dr. Takashi Fujii, Dr. Hiroshi Kawanishi, Mr. Nobuyuki Nishiguchi, of NEC Corporation, Dr. Masao Sato of Takushoku University, Kazuhiro Ueda

of Shibaura Institute of Technology, and Dr. Kenji Ishida of Hiroshima Prefectural University.

Finally, I appreciate the past and present colleagues of the Electronic Circuits and Systems Laboratory for their help and friendship. In particular, I acknowledge Mr. Kazunori Isomoto, Mr. Hiroshi Izumoto, Mr. Chin Saw Kiun Mr. Tetsushi Koide, Mr. Yoshihiro Toyohara, and Mr. Kazutoshi Yokoyama for their programming of algorithms.

TABLE OF CONTENTS

LIST OF SYMBOLS

$x_i \in \mathbf{X}$:	x_i is an element of set X.
$\mathbf{X} \subseteq \mathbf{Y}$:	A set X is contained in set Y.
$\mathbf{X} \cup \mathbf{Y}$:	Union of sets X and Y.
$\mathbf{X} \cap \mathbf{Y}$:	Intersection of sets X and Y.
$\|\mathbf{X}\|$:	Cardinality of set X.
L:	Logic circuit.
M:	A set of modules.
M_i:	A module.
Mh:	A set of hard modules.
Mc:	A set of center moodules.
Mp:	A set of peripheral modules.
Ms:	A set of soft modules.
P:	A set of I/O pads.
N:	A netlist.
n_i:	A net n_i.
G:	A set of groups.
L(M):	A placement of modules.
$\mathbf{L}(M_i)$:	A placement of a module M_i.
$\mathrm{M_i}$:	A set of modules at level i.
$M_j \in \mathrm{M_i}$:	A module M_j in the set $\mathrm{M_i}$.
$\mathrm{R_i}$:	A chip region at level i.

Gg_i: A global routing graph at level i.

Gpv_i: A vertical positioning graph at level i.

Gph_i: A horizontal positioning graph at level i.

LIST OF CAPTIONS FOR FIGURES

LIST OF CAPTIONS FOR TABLES

LIST OF DEFINITIONS

LIST OF EXAMPLES

LIST OF LEMMAS

LIST OF THEOREMS

Chapter 1

Introduction

Over the past three decades, the remarkable growth of the semiconductor industry has been continued, which started from the invention of the transistor, and now realizes an industry that is all-pervasive in its effect on modern life. Each time geometries in the physical design of LSI/VLSI chip are reduced by half, and circuit complexity has potentially increased by four. Moore's law shows a resultant doubling of circuit complexity in each year [Losleben 82].

With the increase of the demand to integrate the whole system into one VLSI chip, Application Specific Integrated Circuit (ASIC) has emerged since early 1980s [Iwata 90]. In addition to the reduction of the design period and the cost, designing the high performance ASIC with millions of gate circuits will require a new Computer Aided Design (CAD) method.

In the following, the layout design method of VLSI circuits with computer aids is briefly introduced.

1.1 VLSI Layout Design

VLSI design consists of the following 6 designs. (1) In functional design, a function specification of a VLSI chip is made and validated. (2) In logic design, this specification is represented by a gate-level circuit consisting of logic gates such as NANDs or INVERTERs. (3) In device design, a transistor as a basic element of a circuit is designed. (4) In circuit design, a transistor-level circuit which implements the gate-level circuit is designed. (5) In layout design, a mask pattern is designed based on the logic circuit and the cell library. (6) In testing, test patterns for verifying the function or performance of the chip are produced and applied to the chip.

The layout design is one of the most difficult and time-consuming tasks in the VLSI design, and it has emerged as one of the most important applications of CAD. In order to obtain layouts for very large circuits, various design styles and strategies of VLSI chip have been proposed.

The gate array approach is one of the most widely used layout styles for the automation of the custom chip design. In this approach, devices are placed on the chip in a simple and regular arrangement without wirings to the other devices, so as to be used in common over the various logic functions [Ohtsuki 86]. Each logic function is realized as a cell by simply routing among devices, and a function of the chip is realized by routing among cells. It takes remarkably short time from the beginning of the chip

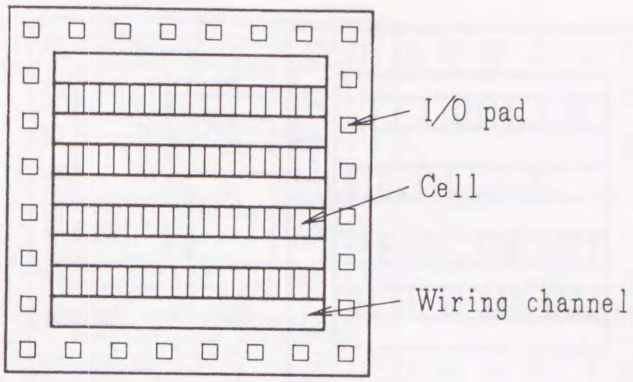


Figure 1.1: Gate array approach.

design to the end of the chip fabrication. But the regular arrangement of cells in this approach usually makes the redundant layout. An example of the chip by the gate array approach is shown in Fig. 1.1.

In the standard cell approach, 30 or 40 types of cells which have the same height are used for various chip design [Ohtsuki 86]. The cells are arranged in rows, but the width of the wiring channel between the rows of cells is flexible so that all the connections can be routed completely. The circuit performance is relatively high, though not to the extent of the building block approach, which is explained next. An example of the chip by the standard cell approach is shown in Fig. 1.2.

The building block approach is one of the most widely used hierarchical layout styles for VLSI layout design. Usually, it is referred to as the general cell or macro cell layout, which is distinguished from the standard cell approach. In this building block approach, the whole circuit on a chip

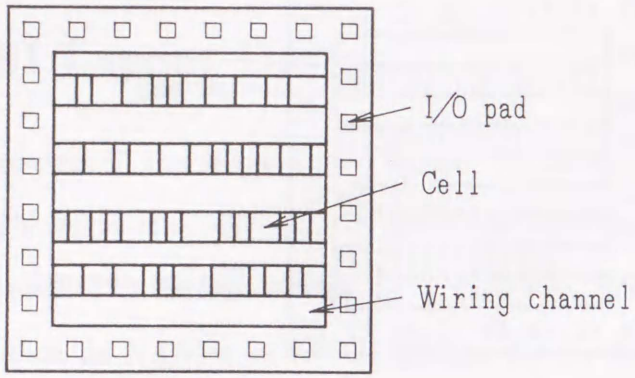


Figure 1.2: Standard cell approach.

is constructed with predesigned functional modules (blocks), which have rectilinear shapes and arbitrary sizes, and a routing region [Kani 83]. In Fig. 1.3, a chip image of the building block approach is shown. In this figure, p_1 through p_8 represent I/O pads, M_1 through M_5 represent modules, and the other area represents the routing region.

In this approach, 100% routability can be always achieved if there is no limitation with respect to the size of routing area [Watanabe 85]. Thus, the most important goal of the layout design in this approach is to obtain the smallest chip in short turn-around-time [Kozawa83], [Markov84].

1.2 Layout in building block approach

The flow of the conventional approach to the VLSI layout design consists of three stages; placement, routing, and layout verification. Routing is divided into consecutive two stages; global routing and detailed routing.

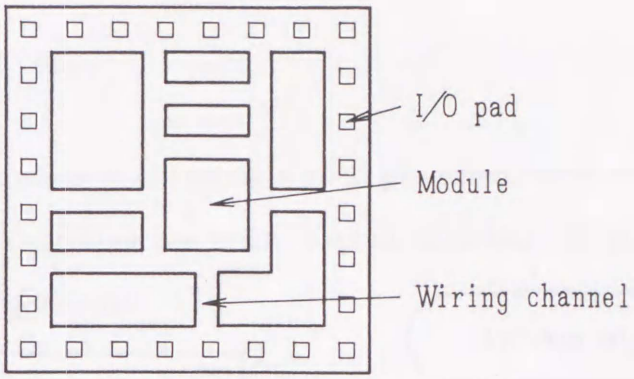


Figure 1.3: Building block approach.

On the other hand, floorplanning has emerged since early 1980s as global placement which is followed by detailed placement (called *positioning*). The details of floorplanning are explained in Chapter 2. (see Fig. 1.4).

A logic circuit, which is an input of the layout design, is usually partitioned into modules in the preceding logic design stage. Then the logic circuit usually consists of a set of *modules* and a *net list.*

In the first stage, modules are placed on a rectangular chip, and the following criteria of the evaluation are adopted.

- Minimization of the estimated wire length (the half perimeter of the minimum enclosing rectangle for a net)
- Minimization of the number of the nets which cross the cut lines
- Minimization of the number of the nets on the cell borders (the chip is divided by the grid lines and each divided region on the chip is

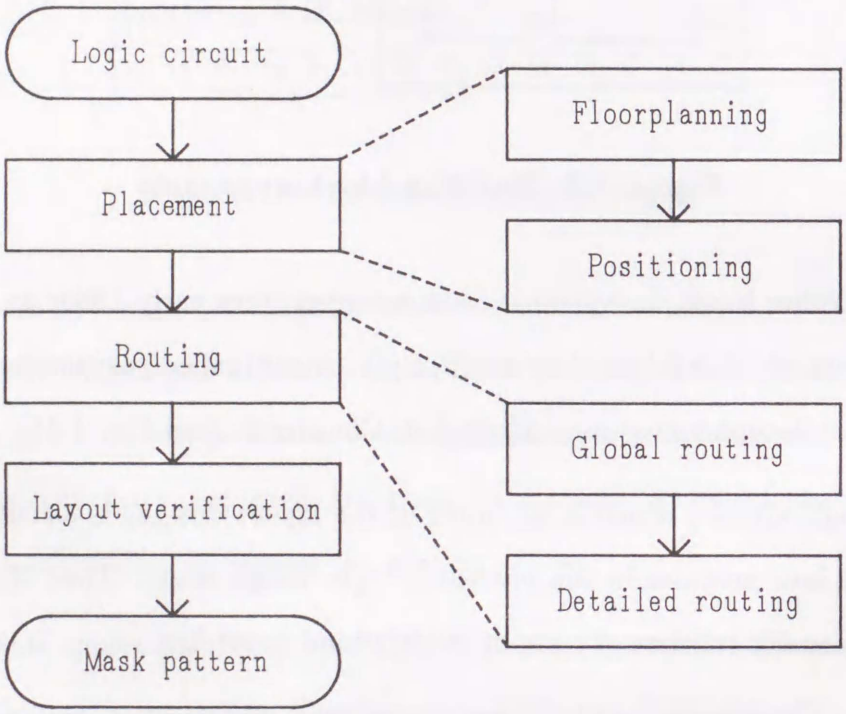


Figure 1.4: VLSI layout design.

called the cell)

Many approaches to obtaining a good placement have been developed. Some of these algorithms are briefly explained in the following.

Constructive algorithms:

- In a typical constructive placement method, modules are selected, one at a time from the modules which are not yet selected, and are placed so that the wire length is minimum. Once a module is placed on a certain position, it is not moved. There is another type of constructive placement which obtains the placement of all modules at a time based on the force model [Ueda 85]. This model adopts attractive force to represent the connections between modules and repulsive force to avoid overlaps.

Iterative improvement algorithms:

Assume that an initial placement of modules is given in advance.

- Pairwise interchange [Watanabe 85]: Two modules are arbitrary selected and they are interchanged. If the placement is improved by this interchange, it is adopted.

- Force directed relaxation [Ohtsuki 86]: One module is arbitrary selected and the median of the module, which is the position that the wire length associated to the module is minimum, is calculated and

the module is placed there. If another module exists on that position, that module is removed and replaced by the same procedure.

- Steinberg method [Watanabe 85]: For a set of modules which are not connected one another, these modules are replaced so that the total wire length is minimum. This problem is resolved by using the maximum matching problem, which is solved in $O(|M|^3)$ by the network flow algorithm, where |M| is the number of the modules.

In general, a constructive algorithm obtains a solution very fast, and an iterative algorithm obtains a good solution.

In the second stage, the interconnections among modules are routed in the routing region. The routing region is usually divided into a set of rectangular regions, called channels and switchboxes, or a set of L-shaped regions [Watanabe 85], [Chen 84]. The terminals around the modules and I/O pads are connected in these regions. This stage consists of the following two stages; *global routing* and *detailed routing.*

Global routing algorithms:

- In the global routing algorithm for building block approach, the channels and the switchboxes are represented by the channel graph [Ohtsuki 86] (see Fig. 1.5). On this graph, Steiner trees for each net are obtained so that the estimated chip size and the total wire length is minimum. The Steiner tree problem is known to be $NP-hard$ [Gary

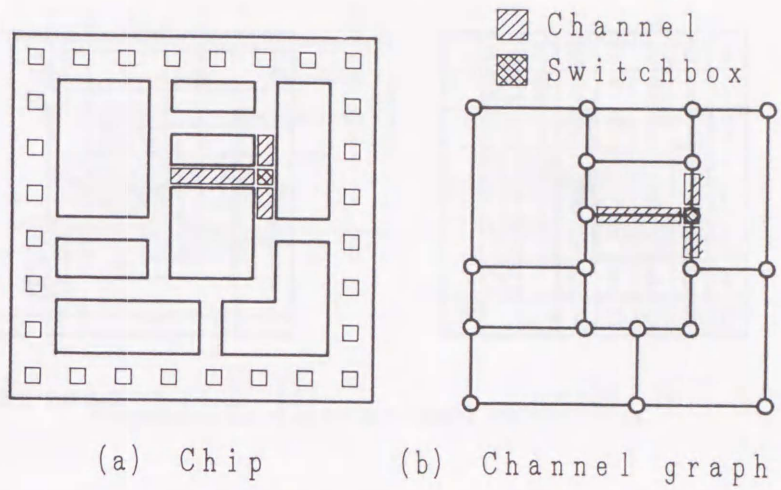


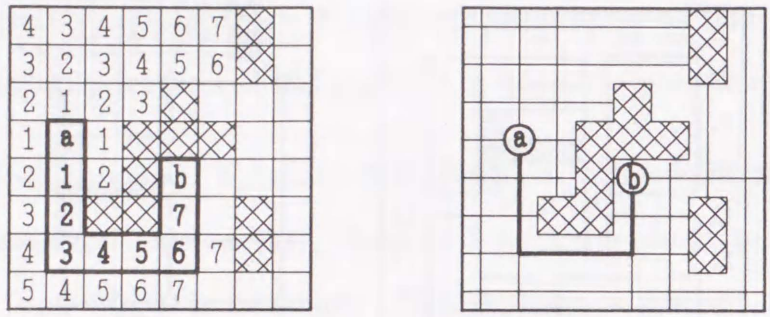
Figure 1.5: Chanel graph.

76], and many heuristic algorithms are proposed [Kou 78] [Ohtsuki 86].

Detailed routing algorithms:

- Maze running algorithm [Ohtsuki 86] (see Fig 1.6 (a)): Let a and b be a pair of terminals of a net. For all cells adjacent to the cell a, the label "1" is attached. Then for all cells adjacent to the cell labeled "1", the label "2" is attached. This procedure is repeated until the label "k" is attached to the cell b. Then the path is found by tracing the label in the decreasing order from b.

- Line-search algorithm [Watanabe 85] (see Fig 1.6 (b)): Let a and b be the terminals of a net located on some intersection point of imaginary



(a) Maze (b) Line-saerch

Figure 1.6: Maze and line-search algorithms.

grids. From a and b vertical and horizontal segments are generated and expanded until they hit obstacles or the chip boundary. From all grid points of the lines, new lines are generated and this procedure is repeated until the line originated from a intersects the line originated from b.

- Channel router [Yoshimura 82] (see Fig. 1.7 (a)): In a channel, terminals of nets are placed on the top and bottom boundaries. All nets in the channel are routed at a time by the left edge method [Yoshimura 82] or dog-leg router [Deutdch 76]. The reduction of the height of the channel is the objective.

- Switchbox router [Ohtsuki 86] (see Fig. 1.7 (b)): In the switchbox, terminals of nets are placed all four boundaries. The objective of this switchbox problem is to achieve 100% routability.

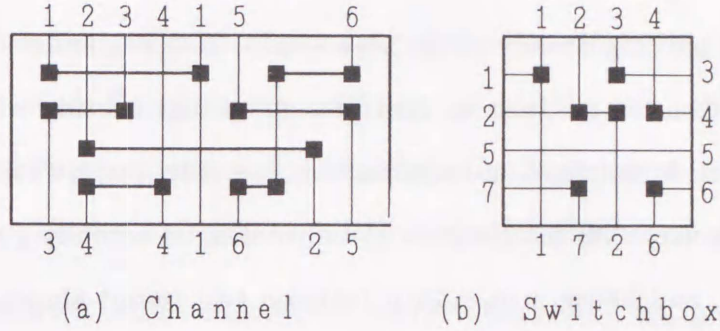


Figure 1.7: Channel and switchbox.

The layout result is usually represented by the geometrical pattern data. In the third stage of the VLSI layout design, these geometrical data is checked so as to have no design errors. In the following, two types of the design errors, which may occur in VLSI layout design, are briefly explained.

(i) Geometrical design error

A particular process technology of VLSI devices has its own geometrical tolerances, such as minimum spacing between shapes, minimum internal width of a shape, and so on. A set of such rules is called the *geometrical design rule*. Violation of the geometrical design rule usually decreases the chip manufacturing yield, or results in non-operational VLSI chips.

(ii) Topological errors, or logical errors. The *topological errors* include wrong electrical connections between circuit elements, and improper structure of circuit elements. These errors are often detected as *logical errors*. In most cases, these errors are fatal.

As circuit scale increases, a *floorplanning* process becomes a key process

of control in packing density of the object chip. A floorplanner determines relative positions of modules so that placement and routing will successfully be done. In conventional approaches, however, precise estimation of routability is very difficult since no global routing is actually given by the floorplanner, and hence a mismatch between two layout stages easily occurs. To alleviate this difficulty, a new floorplanning method which simultaneously determines a floorplan, and *detailed global routes* which directly correspond to switchboxes and channels, is proposed. In addition, because precise estimation of routability is possible in the proposed method, shape and a precise placement of each soft module can be determined simultaneously with a floorplan and global routes [Ohmura 90]. The details of this floorplanning are explained in Chapter 2.

1.3 Organization of Dissertation

This dissertation discusses hierarchical floorplanning for VLSI building block layout, and proposes a new hierarchical floorplanning method combined with global routing and positioning.

In Chapter 1, the outline of the VLSI layout design is discussed as a background of hierarchical floorplanning. In particular, placement and floorplanning in the VLSI layout design are explained in detail.

In Chapter 2, hierarchical floorplanning is explained in detail. First,

conventional hierarchical floorplanning methods are introduced. Then difficulties of these floorplanning methods are pointed out. Finally, a new hierarchical floorplanning method combined with global routing and positioning is proposed, and the outline of this method is mentioned.

In Chapter 3, some theoretical results concerning the proposed hierarchical floorplanning method are discussed. The first result is an initial placement method by the ideal distance, which is newly introduced. The second result is overlap resolution in which relative positions of modules are preserved. This is formulated as an overlap resolution problem and proved to be $NP-hard$. A heuristic algorithm is also given. The third result is improvement of one dimensional module placement. This is formulated as an improvement problem, and for this problem an optimum algorithm and the proof of its optimality are given.

The proposed hierarchical floorplanning consists of the initial floorplanning and the detailed floorplanning. In Chapter 4, algorithms of each phase in the initial floorplanning and the detailed floorplanning are explained with some experimental results, which verify the effectiveness of these algorithms.

In Chapter 5, a prototype system FLORA II, which is constructed based on the proposed hierarchical floorplanning method, is introduced, and some experimental results concerning the hierarchical floorplanning system FLORA II are given. Not only floorplans but also final layouts

after the global routing are compared with the results obtained by a conventional method.

Finally, the conclusions of this dissertation and the future research works are presented in Chapter 6.

Chapter 2

Proposed Floorplanning Method

This chapter describes the proposed floorplanning method for the VLSI layout design. Section 2.1 explains the conventional hierarchical floorplanning methods. In Section 2.2, difficulties of the conventional methods are pointed out. Section 2.3 presents an outline of the proposed method.

2.1 Conventional Methods of Floorplanning

2.1.1 Conventional Floorplanning

Floorplanning is the first stage in the layout of VLSI circuits. In this stage, the relative positions of the modules to be laid out are determined. Algorithms for the floorplanning are classified into the following 3 types. The first type includes algorithms in which modules are regarded as points or squares with the same size [Ueda 85] [Hsu 87] [Kleinhans 88] [Ying 89]. As an example of such algorithms, a semi-automatic VLSI chip floorplan sys-

tem CHAMP [Ueda 85] is briefly introduced. In the paper of CHAMP, the module level layout design in a hierarchical VLSI layout is called as a chip floorplan. Given a set of modules and a netlist, the algorithm determines the center positions of modules, by the *attractive and repulsive force method (AR method)*, and the subsequent module packing process is performed by gradually moving and reshaping modules with chip boundary shrinking.

[**Example 2.1**] An example of the initial placement by AR method is shown in Fig. 2.1 (a). In Fig. 2.1 (b), (c), and (d), the intermediate result 1, the intermediate result 2, and the final result of the floorplanning of the first type are shown, respectively. They are obtained by gradually moving and reshaping modules with chip boundary shrinking. □

The second type of the floorplanning algorithms is based on a rectangular dual [Maling 82] [Leinwand 84] [Ciesielski 87] [Wimer 88] [Lai 88]. As an example of such algorithms, the paper [Maling 82] is introduced briefly in the following. Given a set of modules and a netlist, the algorithm constructs macros and their connections in order that macros and the nets are represented by a planar triangulated graph [Sany 84], where a macro is the term given to an implementation of a high-level function. An important property of the planar triangulated graph is that there exist one or more rectangular duals which are geometric realizations with rectangular areas and satisfy the adjacent condition for each graph edge (see Fig. 2.2). Then for this graph, a rectangular dual which represents the relative positions of

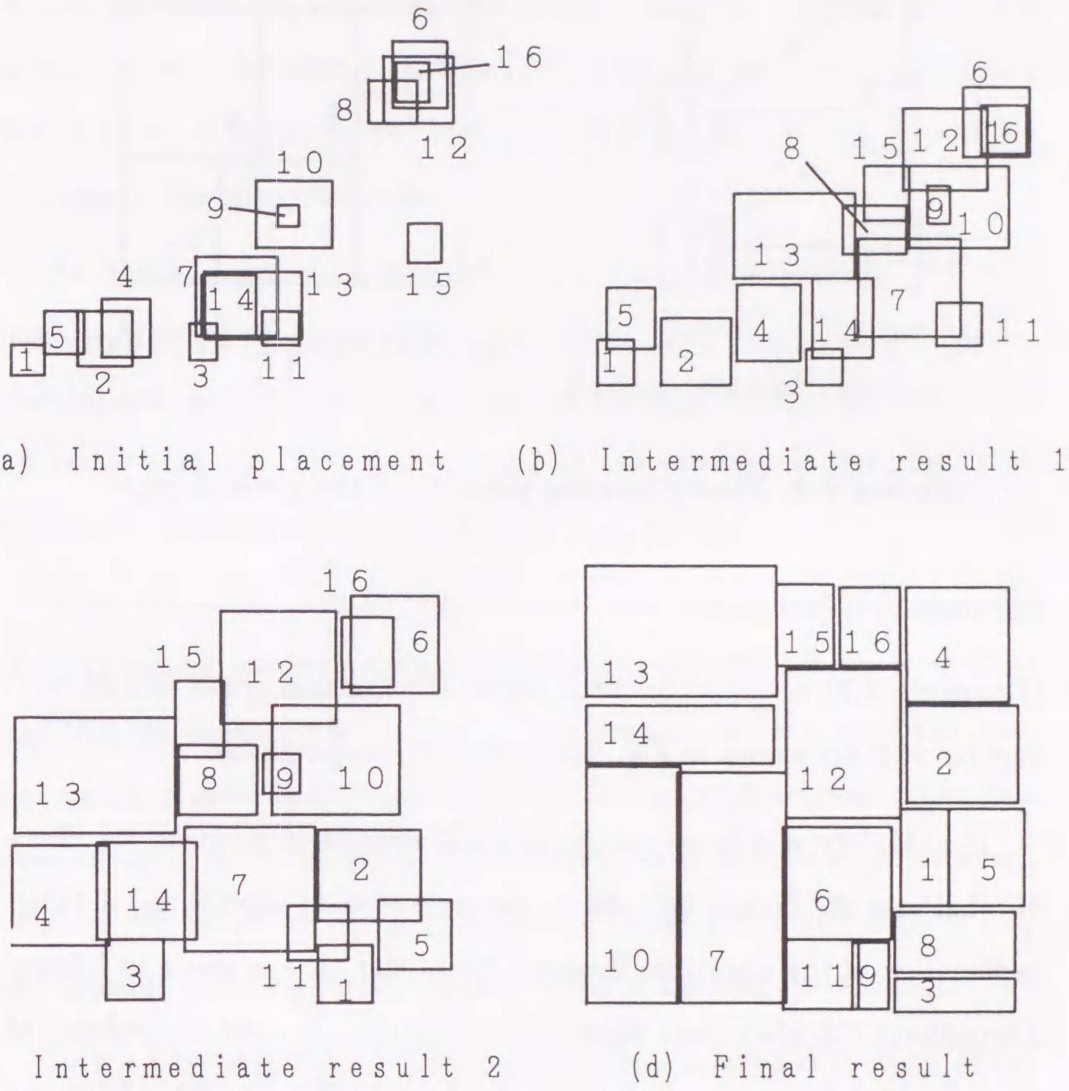
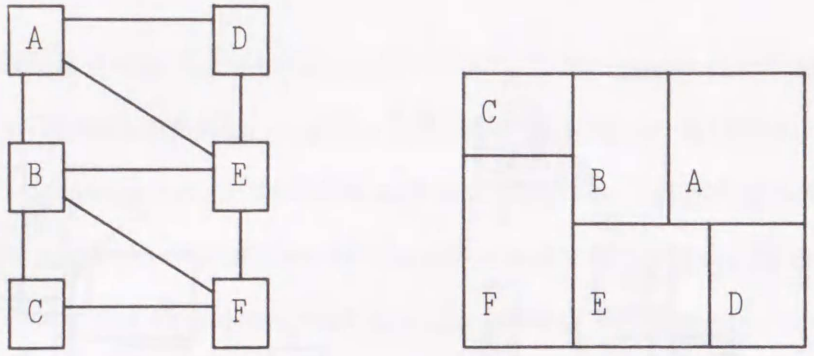


Figure 2.1: Floorplanning process of the first type.



(a) Planar triangulated graph (b) Rectangular dual

Figure 2.2: Floorplanning process of the second type.

the modules is obtained.

[**Example 2.2**] An example of the planar triangulated graph and its rectangular dual are shown in Fig. 2.2 (a) and (b), respectively. □

The third type is based on the min-cut algorithms [Lauther 79] [Tsay 86] [LaPotin 86] [Suaris 89]. As an example of such algorithms, detailed explanation of the algorithm [Lauther 79] of this type is given as follows. Throughout the placement algorithm, the layout of a set of modules M is represented by a pair of mutually dual graphs $Gd_x = (Vd_x, Ed_x)$ and $Gd_y = (Vd_y, Ed_y)$, where Gd_x and Gd_y are planar and acyclic directed graphs containing one source and one sink each (see Fig. 2.3). Parallel edges are also permitted. There is one-to-one correspondence between the edges of Gd_x and Gd_y. Each pair of edges (e_i^x, e_i^y) represents a rectangle

with x dimension $l(e_i^x)$ and y dimension $l(e_i^y)$ where $l(e_i)$ denotes the length associated with the edge e. Furthermore, each pair of edges is associated with a subset of modules. The area $l(e_i^x) \times l(e_i^y)$ equals the total area of the modules in the associated subset.

As inputs, a set of modules M, and a netlist N are given. For each module $M_i \in$ M, the area $s(\mathrm{M}_i)$ is specified. When the algorithm starts, the two graphs contain one edge each and this pair of edges represents the set of all modules. The area covered by the modules is assumed to be quadratic. Therefore $l(e_i^x) = l(e_i^y) = \sqrt{\sum_{M_i \in \mathrm{M}} s(M_i)}$.

First, the set of modules is partitioned into two subsets in such a way that the number of nets incident to modules in different subsets is minimal and that the difference between total module area in the two subsets does not exceed a predefined threshold value. In the graph representation this step corresponds to a splitting of the edge pair into two new edge pairs each of which represents one of the two subsets. The lengths of the edges in Gd_x are adjusted according to the total cell area in the respective subset.

In the next step the partitioning procedure is applied to both of the subsets, then the direction of the cut line is changed and the edge lengths in Gd_y are adjusted.

These steps are repeated hierarchically until each edge pair represents a primitive module.

[**Example 2.3**] A chip and its graph representation are shown in Fig. 2.3

(a) and (b). In Fig. 2.3 (c) and (d), an example of slicing floorplans and its graph representation are shown. □

2.1.2 Hierarchical floorplanning with global routing

In order to obtain a small layout after the routing phase, a floorplanning method with global routing is proposed in [Dai 87] [Dai 89]. In the following, this method is briefly introduced. In this paper, a rectangular floorplan is represented by a rectangular dissection D (see Fig. 2.4 (a)). A rectangular dissection D can be represented by a graph, called the *floorplan graph* Gf = (Vf, Ef), where Vf is a set of vertices representing the intersection of D and there is an edge (v_i, v_j) in Ef if and only if the intersection corresponding to v_i and the intersection corresponding to v_j are adjacent. The inner faces of Gf are called *rooms* (see Fig 2.4 (b)).

To represent the adjacency relations between rooms of Gf, the inner dual graph Gf* = (Vf*, Ef*) of Gf is introduced, where Vf* is a set of vertices representing rooms of Gf, and there is an edge $(v_i^*, v_j^*) \in Ef^*$ if the room corresponding to v_i^* and the room corresponding to v_j^* share a common edge (see Fig. 2.4 (c)).

Since the inner dual graph of a floorplan graph represents the topology of a floorplan, the global routing is executed on connected subgraphs of the inner dual graphs. Such subgraph is called global routing graph.

In this method, global routing is executed hierarchically so that the

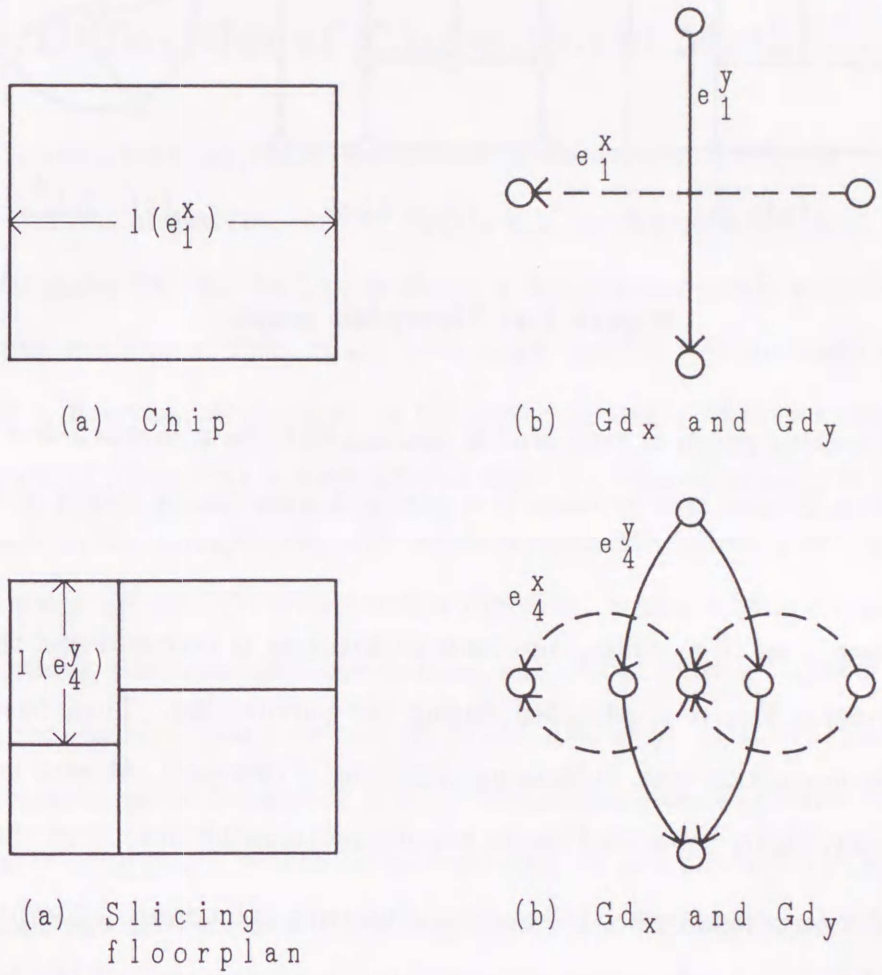


Figure 2.3: Floorplanning process of the third type.

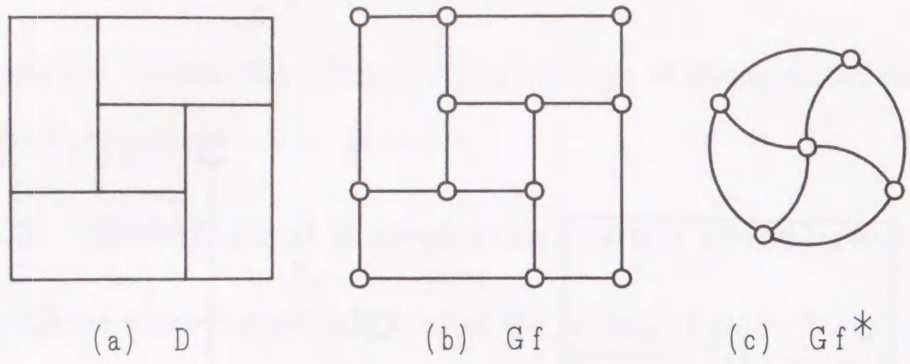


Figure 2.4: Floorplan graph.

global routing graph at each level is guaranteed to be a partial 3-tree. The minimum Steiner tree problem in a partial 3-trees can be solved in linear time.

Given a set of modules, top-down partitioning is executed and the decomposition tree is constructed during the partitioning. Then based on the decomposition tree, bottom-up clustering is executed. At each level of hierarchy, highly connected blocks are grouped together into large clusters.

After forming a tree by hierarchical clustering of modules, a floorplanner and global router together perform a top-down traversal of the tree as follows. Given an overall aspect ratio goal and I/O pad goal, at each level of the tree, the floorplanner searches a simple library of floorplan templates and considers all possible room assignments which meet the combined goals of aspect ratio and I/O pads.

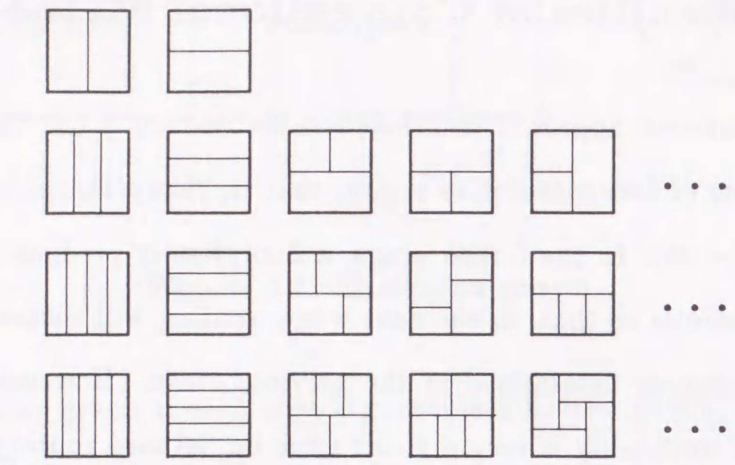
[**Example 2.4**] Floorplan templates are shown in Fig. 2.5 (a), and a

hierarchical structure is illustrated in Fig. 2.5 (b). □

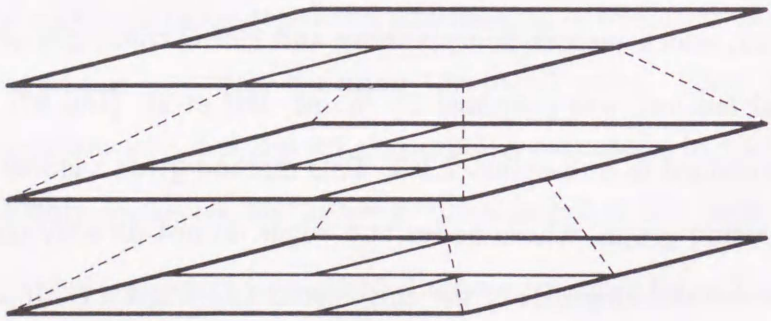
2.2 Difficulties of Conventional Methods

In the conventional approach mentioned in Subsection 2.1.1, the layout design consists of two consecutive stages, that is, *floorplanning* and *routing* [Watanabe 85]. In the former stage, a floorplanner predicts routability among *modules* so that, in the next stage, routing will successfully be done on a floorplan determined in the previous stage. However, precise estimation of routability is very difficult since no detailed routing is actually given by the floorplanner, and hence a mismatch between two layout stages easily occurs. To alleviate this difficulty, a hierarchical floorplanning method, which merges floorplanning and global routing together in a hierarchical fashion, was proposed by W.-M. Dai et al. [Dai 87] [Dai 89], which is explained in Subsection 2.1.3. This method gives a global route on a global routing graph, whose nodes and edges do not directly correspond to switchboxes and channels of the chip layout (see Fig.2.4 (a)), and hence the difficulty mentioned above is partly resolved, but still remains.

In this dissertation, a new floorplanning method which simultaneously determines a floorplan, and *detailed global routes* which directly correspond to switchboxes and channels, is proposed [Ohmura 88]. In addition, because precise estimation of routability is possible in the proposed method, shape and precise placement of each soft module can be determined simultane-



(a) Floorplan templates



(b) Hierarchical structure

Figure 2.5: Floorplan templates.

ously with floorplan and global routes [Ohmura 90].

[**Example 2.5**] A floorplan of the proposed hierarchical floorplanning method is shown in Fig. 2.6. In this figure, the conventional slicing floorplan, which represents only the relative positions of modules, is shown in addition to the floorplan of the proposed method, which consists of the placement (the shapes and the positions) of the modules and the detailed global routes. □

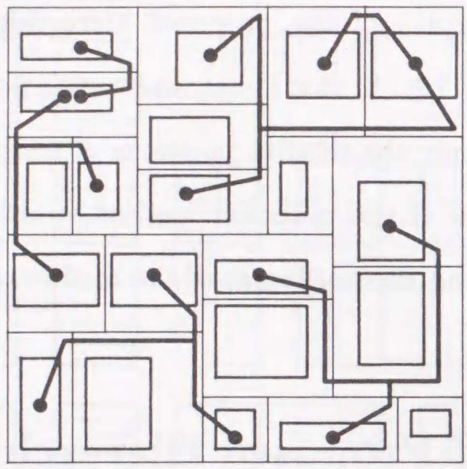
2.3 Outline of Proposed Hierarchical Floorplanning Method

2.3.1 Logic circuit L

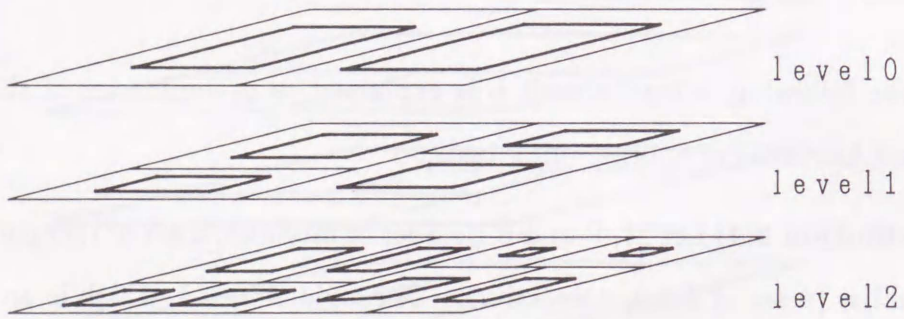
In the following, a logic circuit L is explained as preliminaries of the proposed hierarchical floorplanning method.

[**Definition 2.1**] Let M, P and N be a set of modules, a set of I/O pads and a netlist (a set of nets), respectively. The logic circuit L which is an input of the proposed hierarchical floorplanning is defined by 4-tuple L=(M, P, N, G), where

1. a set of modules M consists of a set of *hard modules* Mh and a set of *soft modules* Ms (M = Mh ∪ Ms), and a set of hard modules Mh consists of a set of *center modules* Mc which have week connectivity to the I/O pads and a set of *peripheral modules* Mp which have strong



(a) Floorplan of the proposed method



(b) Hierarchical structure

Figure 2.6: Floorplan of the proposed method.

connectivity to the I/O pads (Mh= Mc ∪ Mp),

2. for each I/O pad $p_i \in$ P, its position on the perimeter of the chip is specified,

3. a set of *soft modules* Ms is partitioned into a set of groups G = { $G_i \subseteq$ Ms | $1 \leq i \leq k$ } based on their functions, and

4. for each *hard module* $Mh_i \in$ Mh, the shape (specified by a *convex rectilinear polygon*, defined later in this section) is given, whereas for a *soft module* $Ms_j \in$ Ms, only the area s(Ms_j) required to implement Ms_j on the chip is specified. □

[**Example 2.6**] An example of a logic circuit L = (M, P, N, G) is shown in Fig. 2.7, where M = Mh ∪ Ms, Mh = Mc ∪ Mp, Mc = $\{Mc_1, Mc_2\}$, Mp = $\{Mp_i \mid 1 \leq i \leq 4\}$, Ms = $\{Ms_i \mid 1 \leq i \leq 23\}$, N = $\{n_i \mid 1 \leq i \leq 55\}$, G = $\{G_1, G_2, G_3\}$, G_1 = $\{Ms_4, Ms_5, Ms_6, Ms_7, Ms_8, Ms_{19}, Ms_{20}, Ms_{21}, Ms_{23}\}$, G_2 = $\{Ms_1, Ms_9, Ms_{10}, Ms_{11}, Ms_{12}, Ms_{15}, Ms_{16}, Ms_{17}\}$, and G_3 = $\{Ms_2, Ms_3, Ms_{13}, Ms_{14}, Ms_{18}, Ms_{22}\}$, In this figure, a soft module Ms_i is denoted by a circle with its identification number. □

In the design of a VLSI chip, it is useful to utilize modules in an existing VLSI chip which was already designed. In the proposed floorplanning method, these modules can be regarded as hard modules, whose shape may be not only a rectangle but also a *convex rectilinear polygon* defined as follows.

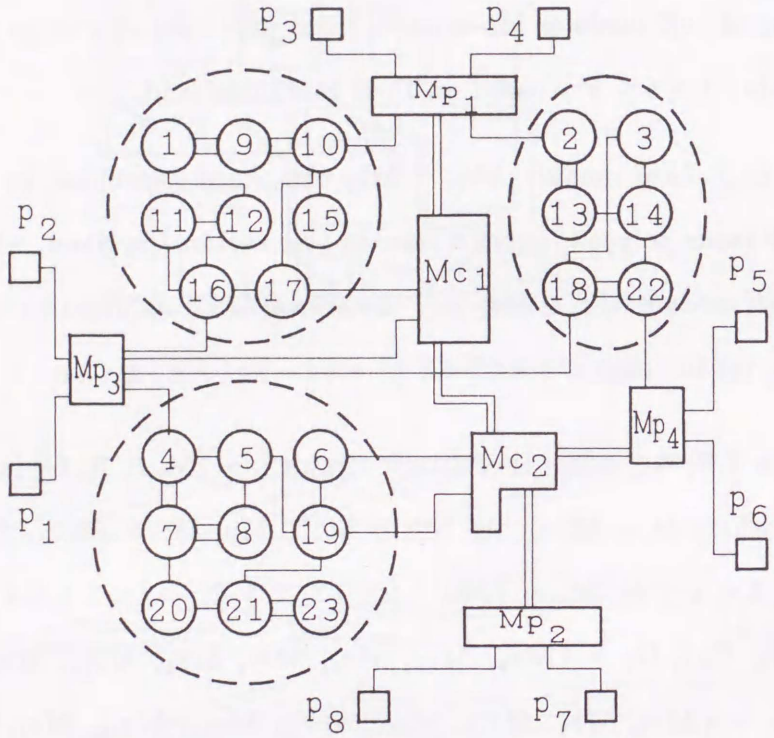


Figure 2.7: Logic circuit L.

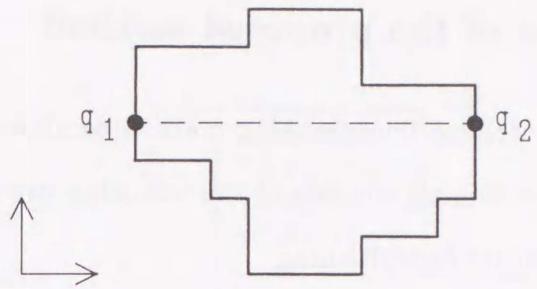


Figure 2.8: Convex rectilinear polygon.

[**Definition 2.2**] A *convex rectilinear polygon* is a rectilinear polygon RP such that for any two points q_1 and q_2 on RP which have the same x or y coordinates, the segment q_1q_2 is entirely contained in the region surrounded by RP (see Fig. 2.8). □

2.3.2 Problem formulation

For a module M_i, a placement L(M_i) is defined by the set of coordinates of the corners. For a set of modules M, a placement L(M) is defined by the set of L(M_i) ($M_i \in$ M). A new floorplanning problem combined with global routing and positioning is formulated as follows.

[**Floorplanning problen FP**] Given a logic circuit L = (M, P, N, G), find a placement of hard modules P(Mh), a placement of soft modules and their shape P(Ms), and global routes T(N) so that the following objective function is minimized.

Objective function: Z = s(chip) (area of the chip)

2.3.3 Outline of the proposed method

The proposed hierarchical floorplanning method combined with the global routing and the positioning consists of the following two stages.

Stage 1: The initial floorplanning

Stage 2: The detailed floorplanning

In Stage 1, the placement of the hard modules and the center positions of the virtual modules are determined. In Stage 2, the virtual modules are repeatedly partitioned into new virtual modules. The global routes of the nets are also determined in this stage. The flow chart of the proposed floorplanning method is shown in Fig. 2.9.

The initial floorplanning (Stage 1) consists of 4 phases. First, a set of virtual module Mg is constructed from Ms based on the group G, where a virtual module is a unit to place. Then all hard and virtual modules are regarded as points, and the positions of these modules are determined in a hypothetically defined chip boundary using the ideal distance. Then the shapes for hard modules are given, and the orientation of each hard module is determined so that the total wire length is minimized. In this phase, overlaps among modules may occur. Then these overlaps are resolved while preserving relative positions of modules. Finally, the placement of hard modules is improved in order to provide the shorter routes (see Section 4.1).

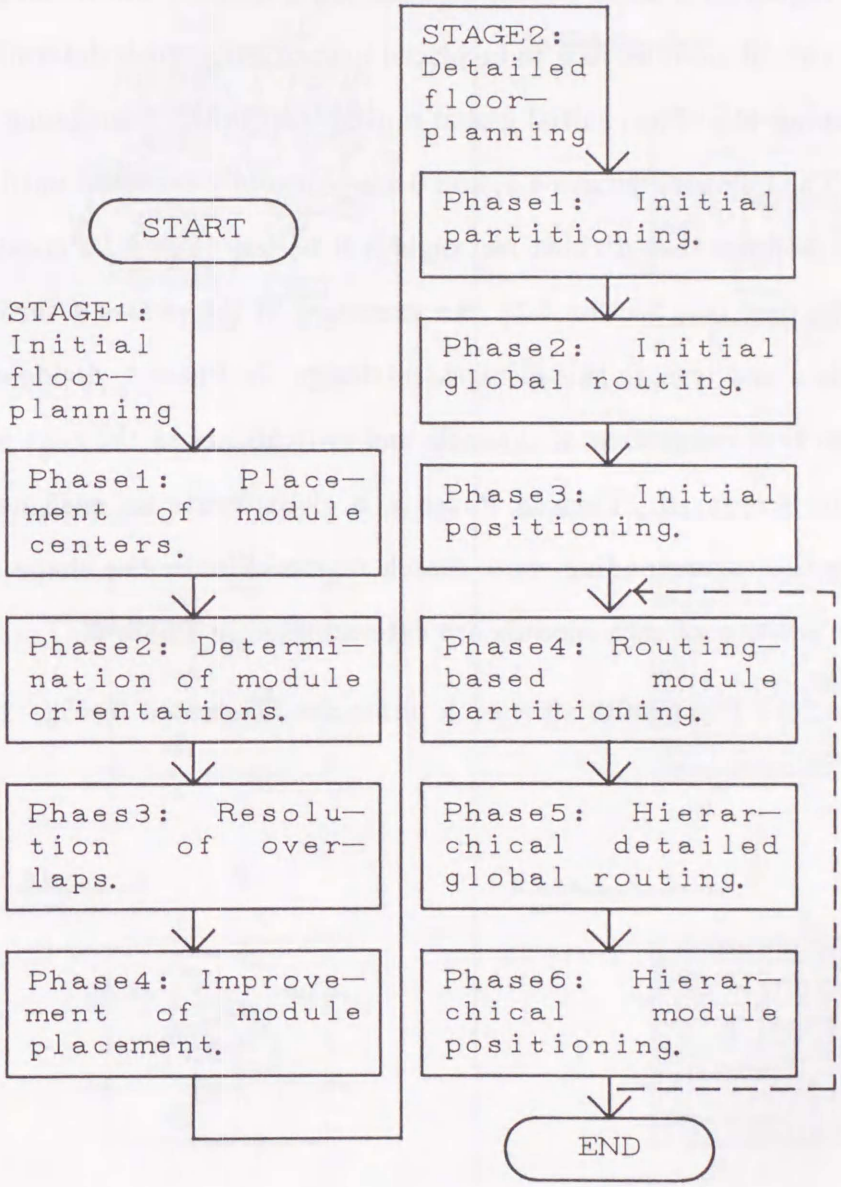


Figure 2.9: The proposed floorplanning method.

The detailed floorplanning (Stage 2) consists of 6 phases. First, the rectilinear region for a set of virtual modules Mg is divided into rectangles, and a new virtual modules Mv_i to be placed in each rectangle is determined by partitioning Mg. Then initial global routing and initial positioning are executed. The following phases 4,5, and 6 are repeatedly executed until the number of modules in a divided rectangle will be less than C, a constant given by the user (see Section 4.2). An execution of the phases 4 through 6 constructs a new level in this hierarchical design. In Phase 4, divide each rectangle so that congestion of channels and switchboxes in the next level of hierarchy is reduced. Then in Phase 5, a global route for each net is obtained in the corresponding route search region. Finally the shape and the precise position of each module are determinated in Phase 6.

[**Example 2.7**] The results after each phase are illustrated in Fig. 2.10.

□

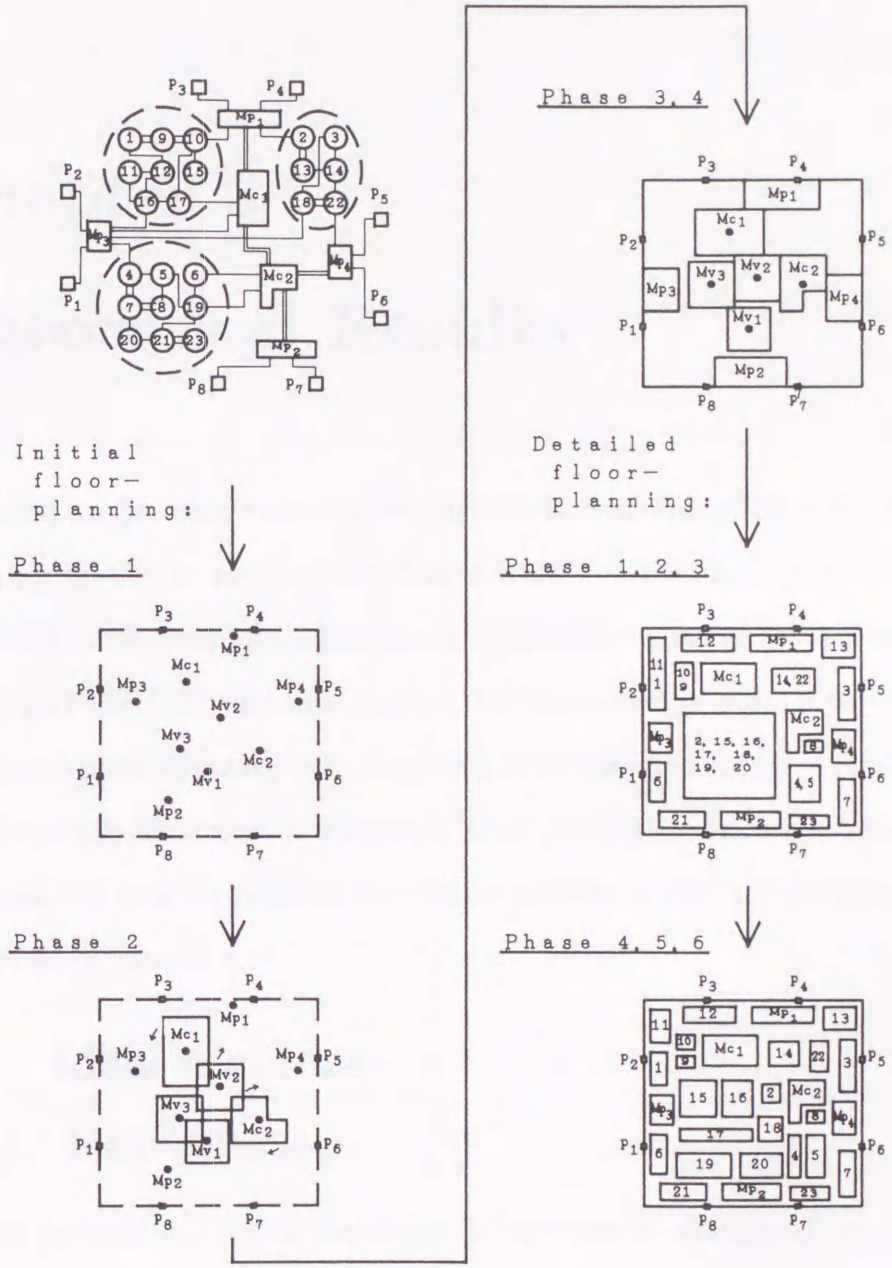


Figure 2.10: The result after each phase.

Chapter 3

Theoretical Results

This chapter presents some theoretical results concerning the initial floorplanning process in the proposed hierarchical floorplanning method. Section 3.1 discusses the initial placement of modules by introducing new concept called the *ideal distance*. Section 3.2 discusses the overlap resolution which preserves relative positions among modules. And finally, one dimensional module placement is discussed. Each problem is discussed individually, and the relation of these algorithms and the initial floorplanning are explained in Section 4.1.

3.1 Ideal distance

3.1.1 Ideal distance

In this subsection, initial placement of modules is discussed. In initial placement of modules, usually, the following two stage method is adopted [Yamada 85]. All modules are assumed to have a square shape with the

same areas. First, relative positions of modules which are regarded as points are determined (the *relative placement*). Then modules are assigned to slots where slots are the rooms on the chip which are divided by the grid lines (see Fig. 3.1). In the first stage, several methods based on the physical model are proposed [Ueda 85] [Yamada 85]. In this relative placement, the total wire length cannot be used directly as its objective because all points would be converged into one point if the total wire length was adopted as objective to be minimized. So, its formulation is not given clearly in most conventional methods. In addition to these facts, conventional algorithms [Ueda 85] [Yamada 85] give the repulsive force and attractive force to each pair of modules independently. Then modules are not distributed uniformly in the chip, and unexpected increase of wire length may occur in the subsequent assignment to the slot. In the following, the *ideal distance* is introduced as a new objective for the relative pacement, in which the attractive effect and the repulsive effect are uniformly taken into account. The relative placement method based on the ideal distance is applied to the initial floorplanning. The details are explained in Section 4.1.

The ideal distance is a distance between two modules. It is calculated from the connectivity between modules and the chip size. The placement of modules is changed so that, for each pair of modules, the difference between the ideal distance and corresponding actual distance between two nodes is minimized. Then strongly connected modules are placed close by

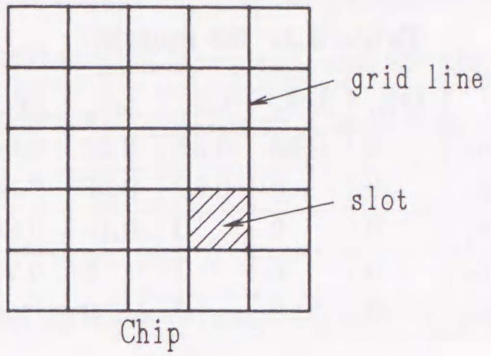


Figure 3.1: Chip model

and weakly connected modules are placed far away on the chip.

First, an undirected graph G=(V, E) for the logic circuit L* for virtual modules M* is introduced.

[**Definition 3.1**] A *connectivity graph* is an undirected graph G=(V, E) such that

- V=$\{M_i \mid M_i \in \mathrm{M}^* \cup \mathrm{P}\}$
- E=$\{(M_i, M_j) \mid M_i, M_j \in \mathrm{V}$, and, $\mathrm{N}_{ij} \neq \phi\}$, where N_{ij} is a set of nets between M_i and M_j (all multi-terminal nets are divided into two-terminal nets in advance), and
- For each undirected edge $(M_i, M_j) \in \mathrm{E}$, a weight $w(M_i, M_j) = \frac{1}{|N_{ij}|}$ is attached. □

[**Definition 3.2**] The *connectivity* b_{ij} between M_i and M_j is defined as

Table 3.1: ID-matrix.

	Mh_1	Mh_2	Mv_1	Mv_2	Mv_3
Mh_1	0	0.35	0.38	0.24	0.54
Mh_2	0	0	0.50	0.36	0.36
Mv_1	0	0	0	0.14	0.86
Mv_2	0	0	0	0	0.72
Mv_3	0	0	0	0	0

the length of the shortest path between two corresponding nodes on the connectivity graph G=(V, E). □

[**Definition 3.3**] The *ideal distance* dp_{ij} between M_i and M_j is defined by multiplying the connectivity between M_i and M_j by a constant C. The constant C is determined so that the length of the longest ideal distance of all pairs of I/O pads is the same as the length of the half perimeter of the chip. A matrix Dp=$[dp_{ij}]$ which represents all ideal distance between any pair of modules is called the *ID-matrix*. □

[**Example 3.1**] The ID-matrix Dp for the logic circuit L* for virtual modules in Fig. 3.1 is shown in Table 3.1. □

A rectangle which satisfies the given aspect ratio and has an estimated area for a given logic circuit is called the *chip region*.

3.1.2 Relative placement problem IDP

By introducing ideal distance, the relative placement problem can be formulated as follows.

[**Problem IDP**] Given a set of modules M* (which are regarded as points) and I/O pads P, the ID-matrix Dp=$[dp_{ij}]$, and the chip region A, obtain the positions of modules in the chip region A so that the following objective function is minimized.

$z = \sum_{for\ all\ ij} |dp_{ij} - dr_{ij}| \quad (i \neq j)$

where dr_{ij} is an actual distance between M_i and M_j (M_i and $M_j \in$ M* $\cup$ P). □

The procedure IDA for the problem IDP is shown as follows.

[**Procedure IDA**]

[**Step 1**] $i \leftarrow j$;

[**Step 2**] For all modules M*, obtain the Manhattan distance dr_{ij} between M_i and all other modules including I/O pads $M_j \in$ M* $\cup$ P;

[**Step 3**] Move module $M_i \in$ M* to the following new coordinates. In the equation, M$^+$ = M* $\cup$ P.

$$x_i' = x_i - \frac{\sum_{M_j \in M+} (dp_{ij} - dr_{ij}) * (x_j - x_i)/dr_{ij}}{|M|-1}$$

$$y_i' = y_i - \frac{\sum_{M_j \in M+} (dp_{ij} - dr_{ij}) * (y_j - y_i)/dr_{ij}}{|M|-1}$$

□

This algorithm can treat modules with different size, very easily. The application to the modules with different size is discussed in Section 4.1.

Initial placement algorithm IP is described as follows.

[**Algorithm IP**]

[**Step 1**] (Initialize) Place all modules in the chip region P by random placement method [Watanabe 85] (I/O pads $p_i \in$ P are fixed around the chip region in advance);

[**Step 2**] (Relative placement) Call the procedure IDA to obtain a relative placement;

[**Step 3**] If the ratio of improvement is more than the given constant C_I, then go to Step 2;

[**Step 4**] (Assignment to the slot) Sort modules on y coordinates, then divide them into groups corresponding to the row of the chip;

[**Step 5**] Sort modules in each row on x coordinates, and assign each module to the corresponding slot; □

The time complexity of the chip is $O(|\mathrm{M}|^3)$.

3.1.3 Experimental results

In order to evaluate the algorithm IP, the algorithm IP is implemented on ECLIPSE MV/4000 of Nippon·Data General with C language. In this experiment, PI method: Pairwise interchange [Watanabe 85], MC method: Min-cut [Watanabe 85], and TP method: Min-cut with terminal propagation [Dunlop 85] are adopted as conventional methods. Data No.1, No.2,

$\cdots$, No.6 are generated randomly by the computer. Data No.8 is a grid data [Ueda 85] and Data No.7 is generated by eliminating edges randomly from Data No.8. Then optimum solutions for Data No.7 and No.8 are known. Table 3.2 (a) shows the input data and the experimental results of the proposed method. Table 3.2 (b) shows the experimental results of the conventional methods for the same data.

From these results, our method decreases the total wire length 6.5% to Pairwise exchange, 28.8% to Min-cut, and 5.3% to Min-cut with terminal propagation.

3.2 Overlap resolution

3.2.1 Relative position

After the initial placement of modules which are regarded as points, the real shapes are given to the modules, and some overlaps among modules may occur. In the conventional layout method, the overlap resolution is not formulated, and the optimization of the placement in the previous phase is not preserved during this step. In this section, the overlap resolution problem which preserves relative positions among modules is discussed. The overlap resolution method which preserves relative positions among modules is applied to the initial floorplanning. The details are explained in Section 4.1.

The shape of a module M_i is assumed to be a convex rectilinear poly-

Table 3.2: Experiments on ideal distance.

(a) Proposed method

Data				*Proposed method*	
No.	$\|M^+\|$	$\|N\|$	*Opt. wire length* (λ)	*Total wire length* (λ)	*CPU time* (sec.)
1	16	43	-	87	6.0
2	16	47	-	92	5.0
3	36	291	-	1057	12.7
4	36	295	-	1071	10.7
5	81	125	-	220	131.7
6	100	100	-	209	108.3
7	96	99	99	218	203.5
8	96	144	144	144	122.1

(b) Conventional method

Data	*Conventional method*					
	PI method		*MC method*		*TP method*	
No.	*Total wire length* (λ)	*CPU time (sec.)*	*Total wire length* (λ)	*CPU time (sec.)*	*Total wire length* (λ)	*CPU time (sec.)*
1	89	7.0	94	1.1	91	1.3
2	90	8.4	98	1.1	93	1.3
3	1042	47.1	1144	5.9	1051	7.0
4	1062	53.0	1155	5.8	1060	7.0
5	220	75.8	429	96.4	219	111.7
6	226	39.2	446	77.8	203	90.0
7	194	82.6	452	77.1	219	89.1
8	318	99.6	630	110.5	272	127.6

gon. L(M_i), a placement of M_i, is defined by the set of coordinates of all corner points of M_i when the module M_i is placed on a xy-plane. L(M), a placement of M, is defined by the set of all L(M_i) ($M_i \in$ M).

For all possible relative positions of a pair of modules M_i and M_j on a placement L(M) are the following 5 cases: (1) "overlap top left of", (2) "overlap bottom left of", (3) "adjacent and left of", (4) "adjacent and below", and (5) "no relation".

First, the center of a module is introduced in order to define the overlap between modules. For each module M_i, consider the point obtained as the average of xy-coordinates of corner points of M_i. If this point is inside of M_i, the center of M_i is defined by this point. If this point is outside of M_i, the center of M_i is defined by the nearest (left most, and bottom most, if necessary) corner points of M_i from this point by *Manhattan distance* [Watanabe 85]. In the following, the center of M_i is denoted by (cx_i, cy_i).

[**Definition 3.4**] For a pair of modules M_i and M_j on L(M), M_i is said to *overlap top left of* M_j if both the conditions 3.1 (a) and 3.1 (b) are satisfied, and M_i is said to *overlap bottom left of* M_j if both the conditions 3.4 (a) and 3.4 (c) are satisfied.

(Condition 3.4)

(a) There exists a line ℓ which satisfies the following 1 - 4.

1. ℓ is a line which is parallel to x axis.

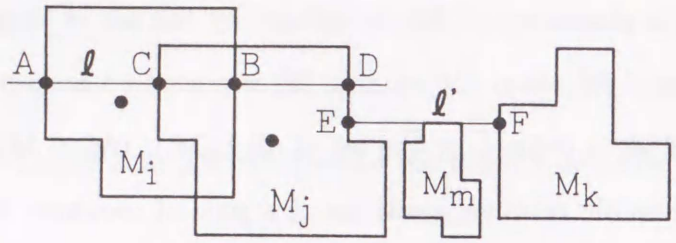


Figure 3.2: Overlapped and adjacent.

2. ℓ crosses at right angle with two edges of M_i (assume that the cross points are A and B, and the x-coordinate of A is less than that of B).
3. ℓ crosses at right angle with two edges of M_j (assume that the cross points are C and D, and the x-coordinate of C is less than that of D).
4. For the cross points A, B, C, and D described in (2) and (3), C is between A and B or A is between C and D.

(b) $[cx_i < cx_j$ or $[cx_i = cx_j$ and $i < j]]$ and $[cy_i > cy_j$ or $[cy_i = cy_j$ and $i > j]]$.

(c) $[cx_i < cx_j$ or $[cx_i = cx_j$ and, $i < j]]$ and $[cy_i < cy_j$ and $[cy_i = cy_j$ and $i < j]]$. □

[**Example 3.2**] An example of overlapped modules M_i and M_j are shown in Fig. 3.2. In this figure, M_i overlaps top left of M_j. □

[**Definition 3.5**] M_j is said to be *adjacent and left of* M_k if a pair of modules M_j and M_k, which are not overlapped each other, satisfies the following condition 3.5.

(Condition 3.5) There exists ℓ' which satisfies the following (1)-(3).

(1) ℓ' is a segment which is parallel to x (y) axis.

(2) ℓ' crosses at right angle with an edge of M_j and an edge of M_k. Assume that the cross points with edge of M_j and M_k are E and F, respectively, and the x (y) coordinates of E is less than that of F.

(3) For arbitrary M_m ($m \neq j, k$) which has the center such that $cx_j < cx_m < cx_k$ ($cy_j < cy_m < cy_k$), ℓ' does not cross the edge of M_m without the parallel edge to $y(x)$ axis, or end point of an edge.

□

Next, adjacency point is defined for a pair of modules which are adjacent each other.

[**Definition 3.6**] For a pair of modules M_j and M_k, assume that M_j is adjacent and left of (below) M_k. Then pay attention to the segment ℓ which is shortest and has minimum y (x) coordinates in all the segments satisfying the condition 3.2. Adjacency point of M_j (M_k) to M_k (M_j) is defined by E (F), which is the intersection point of this segment ℓ' and the edge of M_j (M_k). Intersection points on E and F are denoted by (ax_{jk}, ay_{jk}) and (ax_{kj}, ay_{kj}), respectively. □

[**Example 3.3**] An example of a pair of modules M_j and M_k, which are adjacent, is shown in Fig.3.2. In this figure, M_j is adjacent and left of M_k. E and F are the adjacency points of M_j to M_k and M_k to M_j, respectively.

□

[**Definition 3.7**] For moving operations of each module in L(M), only parallel displacement is allowed. If the following conditions 3.7 (a) and (b) are satisfied, L(M)' is said to *preserve relative positions* in L(M).

(Condition 3.7)

(a) For a pair of modules M_i and M_j in L(M) such that M_i overlaps top left (bottom left) of M_j, the orders on the x axis and y axis of centers are preserved on L(M)'.

(b) For a pair of modules M_i and M_j in L(M) such that M_i is adjacent and left of (below) M_j, the orders on x axis (y axis) of adjacency points are preserved in L(M)'.

□

3.2.2 Problem ORP

A given placement is called the *initial placement* $L_I(M)$, and a placement after the resolution is called the *final placement* $L_F(M)$.

Manhattan distance between centers of M_i and $M_j \in M$ such that M_i overlaps top left or bottom left of M_j in L(M) is denoted by $d(M_i, M_j|L(M))$.

The area of the minimum rectangle which contains all the modules in L(M) is denoted by A(L(M)).

In addition a set of pairs of modules M_i and M_j, such that M_i overlaps top left or bottom left of M_j in L(M), is denoted by OL(L(M)), where OL(L(M)) = $\{(M_i, M_j) | M_i, M_j \in$ M, and, M_i overlaps top left or bottom left of $M_j\}$.

[**Definition 3.8**] The *Moving distance* (Manhattan distance) from L_I(M) to L_F(M) is defined as follows.

$D(L_I(M), L_F(M)) = \sum_{(M_i, M_j) \in OL(L_I(M))} \mid d(M_i, M_j | L_F(M))$

$- d(M_i, M_j |\ L_I(M)) \mid$ □

A problem of resolving overlaps of convex rectilinear modules is defined as follows.

[**Problem ORP**] Given a set of modules M (convex rectilinear polygon), L_I(M) in which overlaps of modules exist and, two positive constants c_1 and c_2, obtain L_F(M) which satisfies the following condition (i) and (ii), and minimize $z = c_1 \cdot D(L_I(M), L_F(M)) + c_2 \cdot A(L_F(M))$.

(Condition i) $OL(L_F(M)) = \phi$

(Condition ii) L_F(M) preserves relative positions in L_I(M). □

L_F(M), which is a solution of the problem ORP, preserves relative positions in L_I(M) and the sum of the moving distance from L_I(M) to L_F(M) and area of L_F(M) is small. Then the optimization of area and relative positions among modules is reflected to the final placement.

3.2.3 NP-hardness of ORP

The computational complexity of the problem ORP is given as follows. In the proof of $NP-hardness$ of the problem ORP, the problem 3-PAT, which is known to be NP-complete, is shown to be reducible to the decision problem DR corresponding to the subproblem of the problem ORP.

[Problem 3-PAT] A positive integer E, a set F = $\{f_k | 1 \leq k \leq 3m\}$, size $s(f_k)$ for each $f_k \in F$, where $s(f_k)$ is a positive integer, $E/4 < s(f_k) < E/2$, $(\sum_{f_k \in F} s(f_k) = mE)$, are given as inputs. Is there partition $F_1, F_2, \cdots, F_m$, $|F_y| = 3$ $(1 \leq y \leq m)$ of F which satisfies the following condition P1 ?

(Condition P1) $\sum_{f_k \in F_y} s(f_k) = E$ $(1 \leq y \leq m)$ □

As a subproblem of the problem ORP, consider the problem in which the shape of modules is restricted to rectangles and the conditions that c_1 = 0, c_2 = 1 are added. The decision problem DR for this problem is defined as follows.

[Decision problem DR] A set of rectangular modules M, initial placement of modules L_I(M), constant C, are given as inputs. Is there final placement L_F(M) of M which satisfies the following conditions D1 - D3 ?

(D1) OL(L_F(M)) = ϕ

(D2) L_F(M) preserves relative positions in L_I(M)

(D3) A(L_F(M)) ≤ C □

In the following, the width and the height of a module M_i are denoted by $w(M_i)$ and $h(M_i)$, respectively. For simplicity, the placement of M_i is

denoted by $\mathrm{L}(M_i) = (x_i, y_i)$, utilizing the coordinates (x_i, y_i) of the bottom left corner of the rectangular module M_i.

[Lemma 3.1] The decision problem DR is $NP - complete$.

(Proof) It is trivial that the decision problem DR belongs to the class NP.

The polynomial transformation from the problem 3-PAT to the problem DR is shown as follows. As an instance, assume that a positive integer E, a set $F = \{f_k | 1 \leq k \leq 3m\}$ and the size $s(f_k)$ for each f_k are given. Based on this, an instance of the problem DR is constructed as the following (i) - (iii).

(i) Let $\mathrm{M_O}$ and $\mathrm{M_I}$ be $\{M_B, M_T, M_L, M_R\}$ and $\{M_k | 1 \leq k \leq 3m\}$, respectively. Then a set of modules M is constructed by $\mathrm{M} = \mathrm{M_O} \cup \mathrm{M_I}$.

The width and the height of each rectangular module are constructed as follows. $w(M_B) = 16mE$, $h(M_B) = 4mE$, $w(M_T) = 16mE$, $h(M_T) = 40mE$, $w(M_L) = 8mE$, $h(M_L) = 4mE$, $w(M_R) = 4(2m-1)E$, $h(M_R) = 4mE$, $w(M_k) = 4s(f_k)$, $h(M_k) = 4E$.

(ii) The initial placement of modules $\mathrm{L_I(M)}$ is constructed as follows (see Fig. 3.3). $\mathrm{L_I}(M_B) = (0,0)$, $\mathrm{L_I}(M_T) = (0, 6mE)$, $\mathrm{L_I}(M_L) = (2mE, 3mE)$, $\mathrm{L_I}(M_R) = (4(m+1)E, mE)$, $\mathrm{L_I}(M_k) = (12mE + \sum_{i<k} 4s(f_i),\ 14mE + 4(k-1)E)$.

(iii) $C = 768\ (mE)^2$.

The above transformation can be executed in polynomial time on |M|. In the following, equivalency of the instance of the problem 3-PAT and the

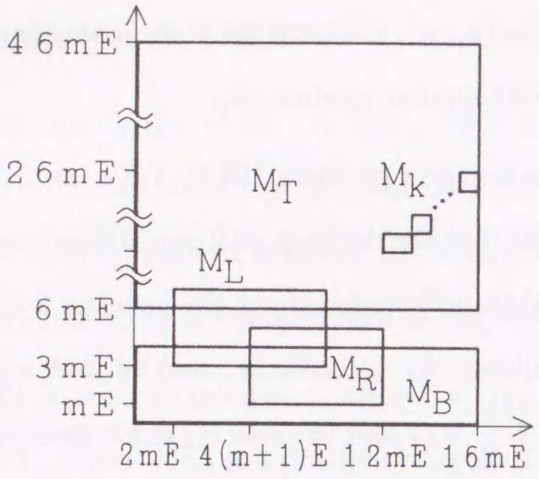


Figure 3.3: Initial placement $L_I(M)$.

instance of the problem DR is proved.

First, assume that the solution of the problem 3-PAT is true. Then, there exists partition of F which satisfies the condition $P1$. Assume this to be $\{F_1, F_2, \cdots, F_m\}$, $F_y = \{ f_{3(y-1)+x} \mid 1 \leq x \leq 3 \}$. Without loss of generality, module $M_{3(y-1)+x}$ corresponding to $f_{3(y-1)+x}$ is denoted by M_y^x. The placement of M_y^x, M_B, M_T, M_L, $and M_R$ is constructed as follows (see Fig. 3.4 and Fig. 3.5, where Fig. 3.5 shows the detailed placement of the region denoted by oblique lines in Fig. 3.4), i.e.:

$L_F(M_T) = (0, 8mE)$, $L_F(M_B) = (0,0)$, $L_F(M_L) = (0, 4mE)$, $L_F(M_R) = (4(2m+1)E, 4mE)$, $L_F(M_y^x) = (8mE + (\sum_{i<x} 4s(f_{3(y-1)+i}), 4mE + 4(y-1)E))$

It is obvious that $M_B, M_T, M_L, and M_R$ are not overlapped each other

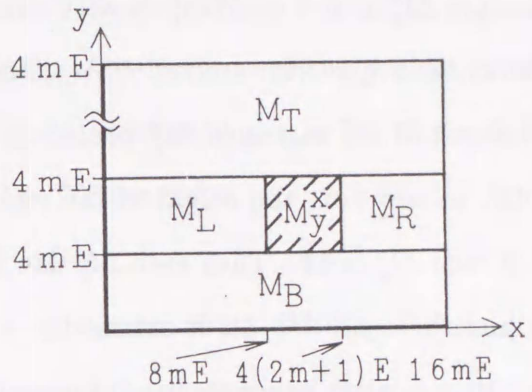


Figure 3.4: Optimum placement of M_B, M_T, M_L, M_R.

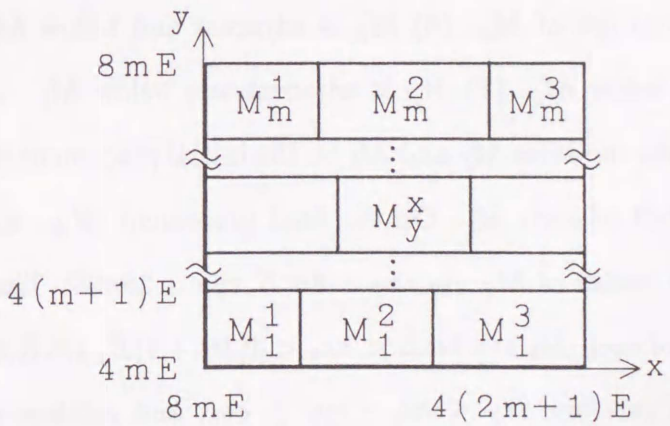


Figure 3.5: Optimum placement of M_y^x.

(see Fig. 3.4). Similarly, each M_y^x is not overlapped (see Fig. 3.4). Next, it is shown that module M_y^x is not overlapped with other modules. The height and the width of the region surrounded by M_B, M_T, M_L and M_R are $4E$ and $4mE$. The width of the region of M_y^x placed as mentioned above is $\sum_{f_k \in F_y} 4s(f_k) = 4E$. In addition, the height of the region of M_y^x is $4mE$ because the height of each M_y^x is $4E$. Then each M_y^x can be placed without overlaps as Fig.3.4, and the condition $D1$ is satisfied.

Next, it is shown that relative positions are preserved. For the relative positions for M_B, M_T, M_L, M_R, and M_k on $\mathrm{L_I(M)}$, only the following (1)-(8) hold. (For $(M_L, M_k), (M_R, M_k), (M_k, M_k')$, relative position does not hold). (1) M_T overlaps top left of each M_k. (2) M_L overlaps top left of M_R. (3) M_L overlaps top left of M_B. (4) M_L overlaps bottom left of M_T. (5) M_B overlaps bottom left of M_R. (6) M_R is adjacent and below M_T. (7) M_B is adjacent and below M_T. (8) M_B is adjacent and below M_k.

Consider the modules M_T and M_k on the initial placement (Fig.3.3). M_T overlaps top left of each M_k. On the final placement (Fig. 3.4), the coordinates of the center of M_T are $cx_T = 8mE, cy_T = 28mE$. The coordinates of the center of each M_k are $8mE < cx_k < 4(2m+1)E$, $4mE < cy_k < 8mE$ (Fig. 3.5). Therefore $cx_T \leq cx_k$; $cy_T \geq cy_k$, and relative positions are preserved. Preservation of relative positions for other modules is shown similarly, and the condition $D2$ is satisfied.

Moreover, the condition $D3$ is also satisfied because $A(L_F(M)) = 768$

$(mE)^2$, $C = 768\ (mE)^2$. Therefore, the solution of the instance of Problem DR is true.

Next, assume that the instance of the problem DR is true. First, it is shown that only the placement shown in Fig. 3.4 satisfies the condition $D1$ - $D3$. Because $C = 768\ (mE)^2$,

$$\mathrm{A}(\mathrm{L}_F(M)) \leq 768(mE)^2 \tag{1}$$

must be satisfied by the condition $D3$. By the condition $D2$, concerning with the coordinates of the centers and the adjacency points,

$$cx_T \leq cx_k \tag{2}$$

$$cy_k \leq cy_T \tag{3}$$

$$cx_L \leq cx_R \tag{4}$$

$$cy_R \leq cy_L \tag{5}$$

$$cx_L \leq cx_B \tag{6}$$

$$cy_B \leq cy_L \tag{7}$$

$$cx_L \leq cx_T \tag{8}$$

$$cy_L \leq cy_T \tag{9}$$

$$cx_B \leq cx_R \tag{10}$$

$$cy_B \leq cy_R \tag{11}$$

$$ay_{RT} \leq ay_{TR} \tag{12}$$

$$ay_{BT} \leq ay_{TB} \text{ and} \tag{13}$$

$$ay_{Bk} \leq ay_{kB} \tag{14}$$

must be satisfied.

First, consider M_B, M_T, M_k which satisfy the condition $D1$. From (12) and (13), M_R and M_B must be placed below M_T on final placement without overlaps. From (16) and (11), regarding with M_R and M_B, two cases, that M_B is placed left of M_R and M_B is placed below M_R, is considered. For the first case,

$$\begin{aligned} h(L_F(M)) &\geq h(M_T) + max\{h(M_R), h(M_B)\} \\ &\geq 44mE, \end{aligned}$$

$$\begin{aligned} w(L_F(M)) &\geq max\{w(M_T), w(M_R) + w(M_B)\} \\ &\geq 4(6m-1)E, \end{aligned}$$

$$\begin{aligned} A(L_F(M)) &= h(L_F(M)) * w(L_F(M)) \\ &= 176(6m-1)mE^2 \\ &> 768(mE)^2 \end{aligned}$$

then, it contradicts the assumption that (1) is satisfied.

On the other hand, for the second case,

$$\begin{aligned} h(L_F(M)) &\geq h(M_T) + h(M_R) + h(M_B) \\ &\geq 48mE(15) \end{aligned}$$

$$\begin{aligned} w(L_F(M)) &\geq max(w(M_T), w(M_R), w(M_B)) \\ &\geq 16mE(16) \end{aligned}$$

$$\begin{aligned} A(L_F(M)) &= h(L_F(M)) * w(L_F(M)) \\ &\geq 768(mE)^2(17) \end{aligned}$$

then, (1) is satisfied if and only if the equal signs in the expressions (15)-(17) are valid (see Fig. 3.4).

Next, consider M_L which satisfies $D1$. M_L must be placed next to M_R, and they must be placed below M_T and above M_B in order to satisfy (1). From (4), M_L can be placed only left of M_R, and M_L, M_R can not be placed outside the width of M_T and M_B (see Fig. 3.4).

Next, consider M_k which satisfies the condition $D1$. From (3) and (14), M_k must be placed in the region below M_T and above M_B, and from (2) and (8), M_k must be placed right of M_L. Now, the width of M_T minus the width of M_L and M_R (i.e. the sum of the width of the region shown by oblique lines in Fig. 3.4) is $4E$. And the height of the region shown by oblique lines is $4mE$. Then, M_k is placed within the region that has the width $4E$ and the height $4mE$. The total area of M_k is $\sum_{f_k \in F} 4s(f_k) * 4mE = 16(mE)^2$. This is the same as the total area of the region $4E * 4mE = 16(mE)^2$. Therefore M_L, M_R, and M_k must be placed between M_T and M_B, and with no dead space. M_L must be placed left of M_R and M_k. Then the left edges of M_L, M_T, and M_B are aligned.

Therefore, only the placement shown in Fig. 3.4 is basically allowed in order to satisfy the conditions $D1$-$D3$. But M_k has the following three ways to be placed into the region shown as oblique line: (1) all M_k are

placed between M_L and M_R (see Fig. 3.4), (2) all M_k are placed right of M_R, (3) M_k are placed left and right of M_R. In the following, the case (1) is discussed, but the cases (2) and (3) can be discussed in the same way.

Based on the placement of Fig. 3.4 (placement of M_k shown by oblique lines), determine the partition of F. From above discussion, M_k are placed with no dead space in the region that the width is $4E$ and the height is $4mE$. Because the height of each M_k is $4E$, and the width is $4s(f_k)$ ($E < 4s(f_k) < 2E$), M_k are placed on m rows. And for each row, just 3 modules are placed. Then, if there are 3 modules b_α, b_β and b_γ in the yth row ($1 \leq y \leq m$), F_y is constructed as $F_y = \{ f_\alpha, f_\beta, f_\gamma \}$. Because $\sum_{f_k \in F_y} 4s(f_k) \leq 4E$, the condition $P1$ is satisfied, and the solution of the problem $3 - PAT$ is true. □

[**Theorem 3.1**] The problem ORP is $NP - hard$. □

3.2.4 Heuristic algorithm ORA

A heuristic algorithm for the problem ORP (Algorithm ORA) is explained.

Two weighted digraph $G_x(L(M)) = (\bar{V}_x, \bar{E}_x)$ and $G_y(L(M)) = (\bar{V}_y, \bar{E}_y)$ are introduced, in order to represent the relation of positions of modules on L(M).

[**Definition 3.9**] Given a placement L(M), the placement graph in x direction for L(M) and imaginary modules M_L and M_R, where M_L and M_R are left to L(M) and right to L(M) respectively, are defined by the following weighted digraph $G_x(L(M)) = (\bar{V}_x, \bar{E}_x)$.

(i) $\bar{V}_x = M \cup \{M_L, M_R\}$

(ii) $\bar{E}_x = \{\overline{(M_i, M_j)} \mid M_i, M_j \in \bar{V}_x$, and, [M_i is adjacent and left of M_j, or, M_i overlaps top left or bottom left of M_j] }

(iii) For each directed edge $\overline{(M_i, M_j)} \in \bar{E}_x$, a weight is attached as a label. This label is represented by $w_x\overline{(M_i, M_j)}$, the value of this is determined as follows.

Method of determination of a label: Consider the line ℓ which is parallel to x axis and intersect the edge of M_i and M_j. Among the edges which intersect the line ℓ, the value of the left most edge on M_j, and the right most edge on M_i are assumed to be $x_\ell(j), and x_\ell(i)$. And $A\ell = x_\ell(j)$ - $x_\ell(i)$ ($A\ell$ takes minus value). Then $w_x\overline{(M_i, M_j)}$ is the minimum value of $A\ell$ for all possible line ℓ. □

[**Definition 3.10**] Given a placement L(M), the placement graph in y direction for L(M) and imaginary modules M_B and M_T, where M_B and M_T are below L(M) and above L(M) respectively, are defined by the following weighted digraph $G_y(L(M)) = (\bar{V}_y, \bar{E}_y)$, like Definition 3.9. □

[**Example 3.4**] An initial placement $L_I(M)$ and introduced four imaginary modules are shown in Fig. 3.6. The corresponding placement graph $G_x(L_I(M))$ is shown in Fig. 3.7 □

Algorithm ORA consists of the following three phases. In Phase 0, for the given initial placement $L_I(M)$, the placement graph and sets OL_x and

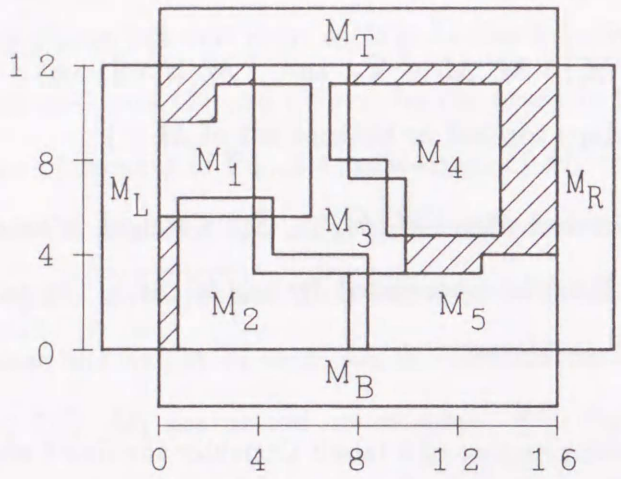


Figure 3.6: Initial placement.

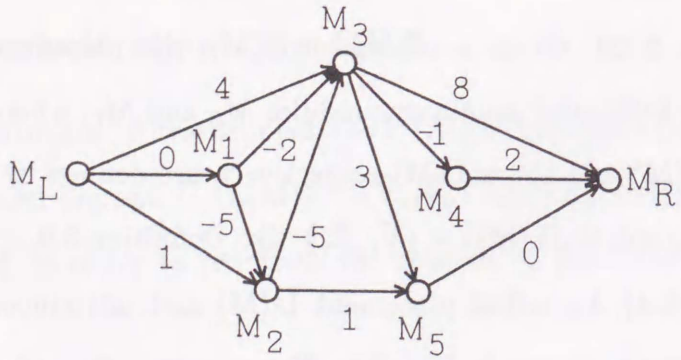


Figure 3.7: Placement graph $G_I(M)$.

OL_y are constructed. OL_x and OL_y are partition of a set OL(L(M)), and they are defined as $OL_x(L(M)) = \{(M_i, M_j) \in OL(L(M)) \mid |w_x(\overline{M_i, M_j})| \leq |w_y(\overline{M_i, M_j})|\}$, $OL_y(L(M)) = OL(L(M))$ - $OL_x(L(M))$.

In Phase 1, overlaps of pairs of modules which belong to OL_x are removed by moving modules in x direction. The obtained placement is called the intermediate placement $L_M(M)$. In Phase 2, overlaps of pairs of modules which belong to OL_y are removed by moving modules in y direction, and obtain final placement $L_F(M)$.

Algorithm ORA is shown as follows.

[Algorithm ORA]

(Phase 0)

[Step 1] For the initial placement $L_I(M)$, construct two placement graphs $G_x(L_I(M)) = (\bar{V}_{Ix}, \bar{E}_{Ix})$, $G_y(L_I(M)) = (\bar{V}_{Iy}, \bar{E}_{Iy})$, and obtain the sets OL_x, OL_y;

(Phase 1 : Resolution in x direction)

[Step 2] Among edges $\overline{(M_j, M_k)} \in \bar{E}_{Ix}$, for all $(M_j, M_k) \in OL_y$, the weight $w_{\ell x}(\overline{M_j, M_k})$ is set to 0 ;

[Step 3] Calculate the length ℓ_{xi} of the shortest path from M_L to each M_i on $G_x(L_I(M))$, and

move each M_i to x direction for the length ℓ_{xi}. The obtained intermediate placement is denoted by $L_M(M)$.

(Phase 2 : Resolution in y direction)

[Step 4] For $L_M(M)$, construct the placement graph $G_y(L_M(M))$.

[Step 5] For $Gy(L_M(M)) = (\bar{V}_{My}, \bar{E}_{My})$ and $Gy(L_I(M)) = (\bar{V}_{Iy}, \bar{E}_{Iy})$, construct the placement graph $G'y(L_M(M)) = (\bar{V}_{My'}, \bar{E}_{My'})$ as follows.

$\bar{V}_{My'} = \bar{V}_{My} = \bar{V}_{Iy}$, $\bar{E}_{My'} = \bar{E}_{My} \cup \bar{E}_{Iy}$. [1]

The weight $w_{My'}(\overline{M_j, M_k})$ is determined as follows.

1. If $\overline{(M_j, M_k)} \in \bar{E}_{Iy}$ - $\bar{E}_{My}$, and $w_{Iy}(\overline{M_j, M_k}) < 0$ then $w_{My'}(\overline{M_j, M_k})$ $= cy_k$ - cy_j
2. If $\overline{(M_j, M_k)} \in \bar{E}_{Iy} \cap \bar{E}_{My}$, and $w_{Iy}(\overline{M_j, M_k}) > 0$, and $w_{My}(\overline{M_j, M_k}) > 0$ then $w_{My'}(\overline{M_j, M_k}) = min\{w_{Iy}(\overline{M_j, M_k}), w_{My}(\overline{M_j, M_k})\}$
3. If $\overline{(M_j, M_k)} \in \bar{E}_{Iy} \cap \bar{E}_{My}$, and $w_{Iy}(\overline{M_j, M_k}) < 0$, and $w_{My}(\overline{M_j, M_k}) > 0$ then $w_{My'}(\overline{M_j, M_k}) = min\{cy_k$ - $cy_i, w_{My}(\overline{M_j, M_k})\}$
4. otherwise $w_{My'}(\overline{M_j, M_k}) = w_{My}(\overline{M_j, M_k})$

[Step 6] Calculate ℓ_{yi}, the shortest path from M_B to each M_i on $G'_y(L_M(M))$, move each M_i to y direction for ℓ_{yi}. The obtained placement is the final placement $L_F(M)$; □

[1] In a special case, a direct edge has different direction on $G_y(L_M(M))$ and $G_y(L_I(M))$. In this case, move the module to the place where this has no edge on $G_y(L_M(M))$, then go to Step 4. There is no effect to the computational complexity.

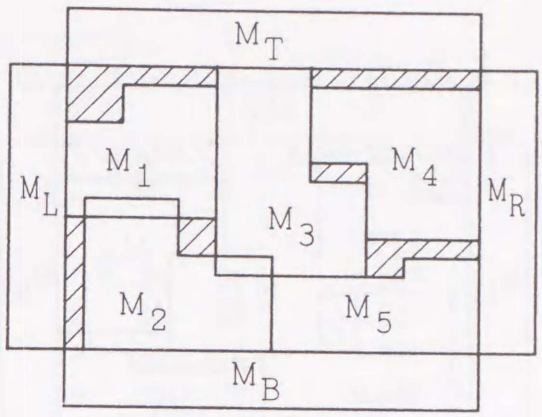


Figure 3.8: Intermediate placement $L_M(M)$.

[**Example 3.5**] Consider the example of the algorithm. In Phase 0, the placement graph $G_x(L_I(M))$ shown in Fig. 3.7, and OL_x and OL_y are obtained. The intermediate placement obtained in Phase 1 is shown in Fig. 3.8. Finally, the final placement $L_F(M)$ obtained in Phase 2 is shown in Fig. 3.9.

Algorithm ORA obtains the feasible solution in $O(m^2)$, where m is the number of nodes, which construct modules, in $L_I(M)$.

In order to evaluate the performance of the proposed heuristic algorithm, the algorithm ORA is implemented in the *C* language on an ECLIPSE MV/4000 (0.6MIPS) of Nippon Data General. The results of the experiments are summerized in Table 3.3 and Table 3.4. Input data is randomly generated both the shape (convex rectilinear polygon) and initial placements for them.

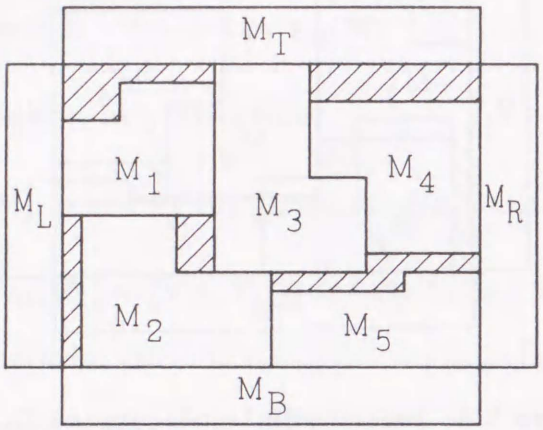


Figure 3.9: Final placement $\mathbf{L_F(M)}$.

Table 3.3: Experimental result 1.

Data			*Opt.*	*Algorithm ORA*		*Ratio*
No.	$\|M\|$	$\|V\|$	Z_0	Z	*CPU time* (sec.)	Z/Z_0
1	3	22	66	70	1.39	1.15
2	4	26	79	101	1.63	1.28
3	4	30	26	28	1.87	1.08
4	5	34	29	38	2.21	1.31
5	5	38	36	39	2.31	1.08

Table 3.4: Experimental result 2.

Data			*Algorithm ORA*
No.	$\|M\|$	$\|V\|$	*CPU time* (sec.)
6	20	98	4.382
7	25	124	5.680
8	30	150	7.797
9	35	178	9.318
10	40	206	9.993

In Table 3.3, the comparison between the solution z obtained by the algorithm ORA and the optimum solution z_0, as an evaluation of the algorithm ORA for small data, where |M| is the total number of modules, and |V| is the total number of nodes which construct the modules. z and z_0 are the value of objective function, and the optimum solution obtained by branch and bound method, respectively, when $c_1 = 1$, $c_2 = 1$. From Table 1, the difference from the optimum solution is within only 18% .

In Table 3.4, the cpu time of the algorithm ORA for practical data is shown. |M| and |V| are the same as those in Table 3.3. For 40 modules, the solution is obtained within 10 seconds, that is a practical time.

From the result of experiments, the algorithm ORA is considered to be an efficient heuristic algorithm.

3.3 One Dimensional Module Placement

3.3.1 Problem MPP

In VLSI layout design for building blocks, to place I/O modules which are placed in the peripheral area of the chip optimally is important to obtain a small chip. In this subsection, an improvement of the placement of those I/O modules is discussed. The algorithm for this problem is applied to the initial floorplanning. The details are explained in Section 4.1.

As a basic research for the improvement of the placement, the following assumptions are made. The shape of all peripheral modules is a rectangle, and those modules are placed on the x axis. In the following, these peripheral modules are simply called modules and denoted by M. The positions of other modules and I/O pads are fixed, and these fixed terminals are called *external terminals*. A set of external terminals is denoted by Te. At first, each net $n_i \in$ N is assumed to consists of two terminals, but it is extended to multi-terminals later in this subsection. A logic circuit LC for the problem in this subsection is represented by 3-tuple LC=(M,Te,N).

In the following, a one dimensional module placement on x axis for the logic circuit LC = (M, Te, N) is considered. When $M_i \in$ M is placed on x axis, the x coordinate of the left side of M_i is represented by $L(M_i)$. Assume that modules $M_i \in$ M are placed according to the increasing order of their subscripts. Then the placement of M is represented by L(M) = $(x_1, x_2, \cdots, x_m)$. The coordinates of the left most side and the right most

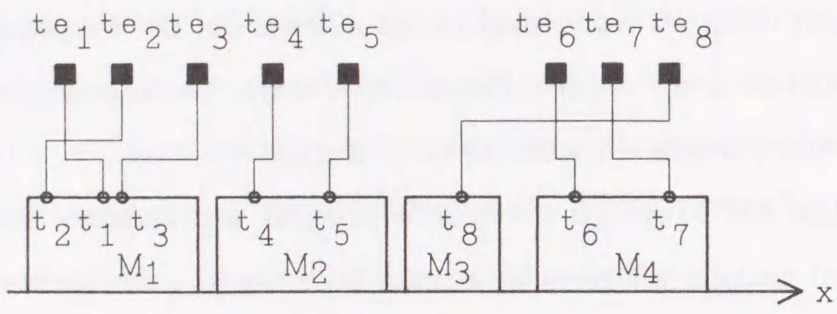
side of L(M) are represented by LB(L(M)) = x_1, RB(L(M)) = $x_m + w(M_m)$. Because one dimensional placement of modules is discussed, the wire length of a net n_i in terms of the half perimeter, which is often utilized in VLSI placement design, is represented by the difference of the x coordinates of two terminals of n_i, and it is denoted by $Zx(n_i)$. The wire length of the net n_i with a weight c_i is denoted by $c_i \cdot Zx(n_i)$.

[**Problem MPP**] A logic circuit LC=(M,Te,N), and an initial placement L_I(M) of modules are given as inputs. Each net $n_i \in$ N consists of two terminals, one of which is a terminal t_j of a module and the another is an external terminal te_j. In the initial placement L_I(M), modules are placed on x axis according to the order of M_1, M_2, $\cdots$, M_m from left. Then find the placement L(M) on x axis which satisfies the following conditions i) and ii), and minimizes the objective function Z=$\sum_{n_i \in N} c_i \cdot Zx(n_i)$, where c_i is the weight of the net n_i and $Zx(n_i)$ is the distance of two terminals on x axis.

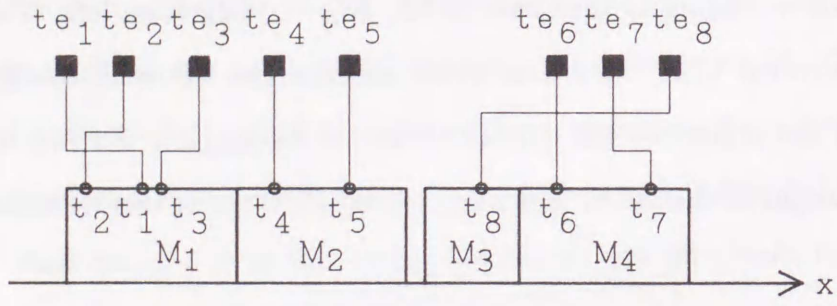
(Condition i) Modules are not overlapped each other.

(Condition ii) Order of modules on the x axis is preserved.

[**Example 3.6**] For an example of the problem MPP, the initial placement L_I(M) with Z= 185 is shown in Fig. 3.10 (a). For this initial placement, the final placement with the objective function Z= 175 is shown in Fig. 3.10 (b). □



(a) Initial placement



(a) Final placement

Figure 3.10: Example of Problem MPP.

3.3.2 Optimum algorithm MPA

In the following, the algorithm MPA for the problem MPP is explained. First, a *concatenated module* which is an important concept to describe the algorithm is explained. A sequence of modules which are placed with no space between them can be regarded as a hypothetical module. This module is called the *concatenated module* and denoted by $CM < j,k > = \{M_j, M_{j+1}, \cdots M_{j+k-1}\}$ if it consists of k modules from M_j. Next, a terminal sorted sequence is explained. $D(te_i)$ is the x coordinate of the external terminal te_i, and $d(t_i)$ is the relative x coordinate of the corresponding terminal t_i of $CM < j,k >$ from its left side. Then l_i for $CM < j,k >$ is defined by $D(te_i) - d(t_i)$. The *terminal sorted sequence* for $CM < j,k >$ is a sequence of l_i which is sorted in nondecreasing order, and is denoted by $l_{<j,k>} = (l_{\pi(1)}, l_{\pi(2)}, \cdots, l_{\pi(\tau)})$, where τ is the number of the nets.

In the following, the placement of a subset M' = $\{M_j, M_{j+1}, \cdots, M_{j+k-1}\}$ $(1 \leq j \leq j+k \leq m)$ of M is denoted by L(M') = $(x_j, x_{j+1}, \cdots, x_{j+k-1})$, where each x_i is the x coordinate of the left side of the module M_i in L(M'). And the objective function related to the nets which connect the modules in M' is denoted by Z(L(M')). The algorithm MPA is shown as follows. The precise description of the procedure PLM and BP in MPA is shown in Fig. 3.11.

[Algorithm MPA]

[**Step 1**] For all $M_i \in$ M, execute the following Step 2 - Step 3.

[**Step 2**] Regard M_i as $CM < i, 1 >$, and construct $l_{<i,1>}$ for $CM < i, 1 >$;

[**Step 3**] Calculate L($CM < r, w >$) by calling the procedure PLM($CM < i, 1 >$) $(r + w - 1 = i)$; □

The algorithm MPA obtains the placement L(M_i) according to the order of M_1, M_2, $\cdots$, M_m, as follows. Consider M_i to be a concatenated module $CM < i, 1 >$. Then calculate the terminal sorted sequence $l_{<i,1>}$ for $CM < i, 1 >$, and call the procedure PLM. PLM($CM < i, 1 >$) returns the placement P($CM < r, w >$) of the concatenated module $CM < r, w >$ $(r + w - 1 = i)$ which includes $CM < i, 1 >$, while repeating the following operations.

In the following the operations in the procedure PLM is explained, first. Then the operations in the procedure BP which is called by the procedure PLM is explained. Given a concatenated module $CM < j, k >$, the procedure PLM returns the placement which minimizes the objective function for $CM < j, k >$ and a set of concatenated modules PM which are already placed. Let us assume that the right most concatenated module in PM to be $CM < r, u >$ $(r+u = j)$. If there does not exist $CM < r, u >$ mentioned above, the procedure PLM determines the placement of $CM < j, k >$ by the procedure BP, and it halts. In the following, the case that there exists $CM < r, u >$ mentioned above is explained.

```
Procedure PLM(CM < j, k >)
  begin
    P(CM < j, k >) ← BP(CM < j, k >);
    /* Let CM < r, u > be the right most
      concatenated module in PM. */
    if LB(P(CM < j, k >)) < RB(P(CM < r, u >)) then
      begin
        PM ← PM − {CM < r, u >};
        Construct CM < r, w > (w=u+k) by concatenating
          CM < r, u > and CM < j, k >;
        Update l<r,w> by merging l<r,u> and l<j,k>;
        P(CM < r, w >) = PLM(CM < r, w >);
        return P(CM < r, w >)
      end
    else
      begin
        PM ← PM ∪ {CM < j, k >};
        return P(CM < r, w >)
      end
end

Procedure BP(CM < j, k >)
  begin
    /* Let l<j,k> = (l_π(1), ···, l_π(r))
      be a terminal sorted sequence of CM < j, k > */
    Obtain the subscript π(q) such that
    c_π(1) + ··· + c_π(q) ≥ (c_π(1) + ··· + c_π(r)) / 2;
     return l_π(q)
  end
```

Figure 3.11: Procedure PLM and BP

First, obtain the placement L($CM < j, k >$) for $CM < j, k >$ by the procedure BP independently of concatenated modules in PM. Then the following two cases occur.

i) LB(L($CM < j, k >$)) < RB(L($CM < r, u >$))

First, remove $CM < r, u >$ from PM. Then obtain the new concatenated module $CM < r, w >$ ($w = u + k$) by concatenating $CM < r, u >$ and $CM < j, k >$. Next, obtain the terminal sorted sequence $l_{<r,w>}$ for $CM < r, w >$ by merging $l_{<r,u>}$ and $l_{<j,k>}$. Finally, obtain the placement L($CM < r, w >$) of $CM < r, w >$ which includes $CM < j, k >$ by repeatedly calling the procedure PLM, and return L($CM < r, w >$).

ii) LB(L($CM < j, k >$)) $\geq$ RB(L($CM < r, u >$))

Add $CM < j, k >$ to PM, and return L($CM < r, w >$).

Next, the operations in the procedure BP is explained. Given a concatenated module $CM < j, k >$, the procedure BP obtains the subscript $\pi(q)$ which satisfies the following condition $C1$, by the terminal sorted sequence. Then return L($CM < j, k >$) = $l_{\pi(q)}$ as the placement of $CM < j, k >$.

[Condition C1] $\pi(q)$ ($1 \leq q \leq \tau$, *where* τ *is the number of terminals of* $CM < j, k >$) is the minimum subscript which satisfies the following inequality.

$$c_{\pi(1)} + c_{\pi(2)} + \cdots + c_{\pi(q)} \geq (c_{\pi(1)} + c_{\pi(2)} + \cdots + c_{\pi(\tau)})/2$$

[Example 3.7] Assume that $M_1, M_2, \cdots, M_7$ are already placed as $CM < 1, 3 >$, $CM < 4, 1 >$, $CM < 5, 3 >$ by Algorithm MPA. And now M_8 is

placed as $CM < 8,1 >$. The weights of nets are c_6=2, c_9=5, c_{12}=c_{13}=3, and 1 for other nets. Figure 3.12 (a) shows the placement P($CM < 8,1 >$) which is determined by the Procedure BP. Since LB(P($CM < 8,1 >$) < RB(P($CM < 5,3 >$), the procedure concatenates these modules and obtains $CM < 5,4 >$ (Fig. 3.12 (b)). Repeat these steps and finally P($CM < 4,5 >$) is obtained as the result of PLM($CM < 8,1 >$) (Fig. 3.12 (c)). □

3.3.3 Optimality of MPA

Next, the time complexity of the algorithm MPA is explained. The procedure BP takes $O(|N|)$ time. The procedure PLM without the procedure BP and the portion which calls PLM itself, takes also $O(|N|)$ time. Because PLM calls BP and PLM itself at most 2 × |M| - 1, whole PLM takes $O(|N| \cdot |M|)$ time. The calculation of terminal sorted sequences takes $O(|N| \cdot log|N|)$. Then the time complexity of the algorithm MPA is $O(|N| \cdot (|M| + log|N|))$.

In the following, the correctness of the algorithm MPA is proved. For L($CM < j,k >$), let α=LB(L($CM < j,k >$)). Then Z(L($CM < j,k >$)) is represented as $f(\alpha)$ by the terminal sorted sequence $l_{<j,k>}$.

$$f(\alpha) = C_{\pi(1)} \cdot |\alpha - l_{\pi(1)}| + \cdots + C_{\pi(r)} \cdot |\alpha - l_{\pi(r)}| \quad \cdots (1)$$

[Lemma 3.2] For the terminal sorted sequence of $CM < j,k >$, there exists π(q) in Condition C1), and for this π(q), $f(\alpha)$ is i) decreasing if $\alpha < l_{\pi(q)}$, ii) nondecreasing if $l_{\pi(q)} \leq l_{\pi(q+1)}$, iii) increasing if $l_{\pi(q+1)} \leq \alpha$.

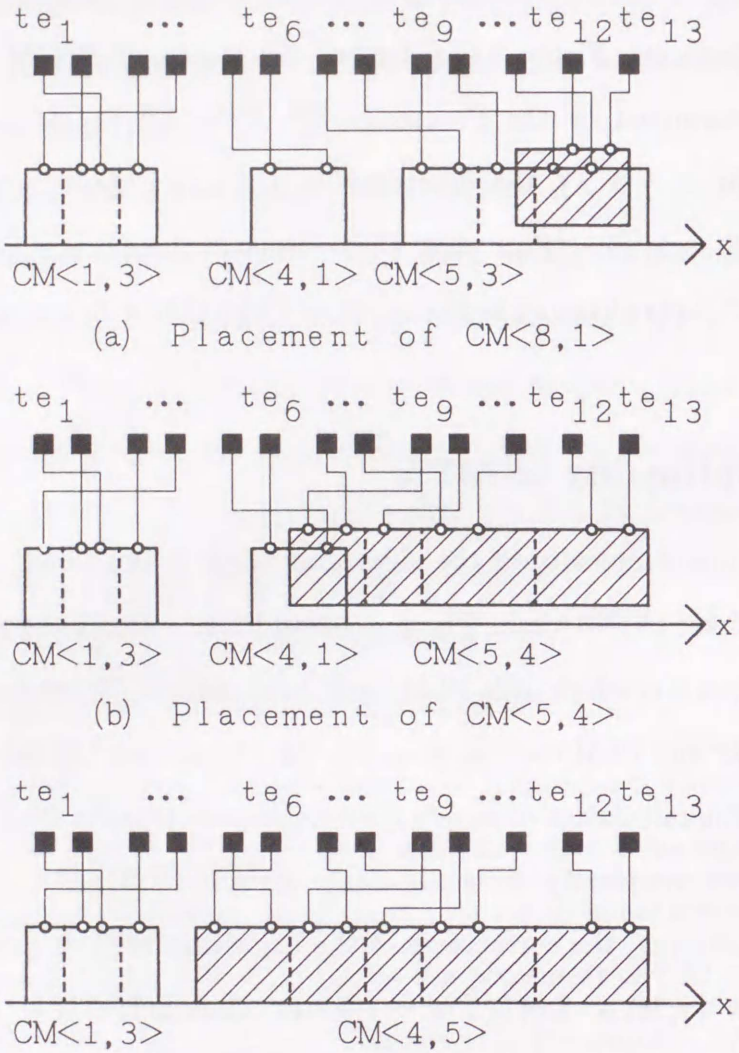


Figure 3.12: Example of the placement.

(Proof) $f(\alpha)$ is represented by the following expressions.

$$l_{\pi(k)} \leq \alpha < l_{\pi(k+1)} \quad (0 \leq k \leq \tau)$$

$$f(\alpha) = (C_{\pi(1)} + \cdots + C_{\pi(k)}) - (C_{\pi(k+1)} + \cdots + C_{\pi(\tau)})\alpha + K_k \quad \cdots(2)$$

($l_{\pi(0)} = -\infty$, $l_{\pi(\tau+1)} = \infty$, K_k is a constant which is determined by the weights of nets, and the positions of terminals of modules and external terminals.)

It is obvious that π(q) which satisfies Condition C1 exists. Next, the following i), ii), and iii) for π(q) are proved.

i) If $\alpha < l_{\pi(q)}$: the inequalities k<q, and $(C_{\pi(1)} + \cdots + C_{\pi(k)}) < (C_{\pi(k+1)} + \cdots + C_{\pi(\tau)})$ hold regarding the coefficient of α in the equation (2). Then $f(\alpha)$ is decreasing.

ii) If $l_{\pi(q)} \leq \alpha < l_{\pi(q+1)}$: the equality k=q, and the inequality $(C_{\pi(1)} + \cdots + C_{\pi(q)}) \leq (C_{\pi(q+1)} + \cdots + C_{\pi(\tau)})\ C_{\pi(\tau)})$ hold regarding the coefficient of α in the equation (2). Then $f(\alpha)$ is nondecreasing.

iii) If $l_{\pi(q+1)} \leq \alpha$: $f(\alpha)$ is increasing because of the same reason of i). □

Lemma 3.2 leads to the following Theorem 3.2.

[**Theorem 3.2**] For $CM < j, k >$, Procedure BP returns the placement L($CM < j, k >$) which minimizes Z(L($CM < j, k >$)).

[**Lemma 3.3**] Assume that the placement L(M) with minimum Z(L(M)) is given for a set of modules M = $\{M_1, M_2, \cdots, M_u\}$, where u $\geq$ 1. If modules are concatenated and placed, the following i) and ii) hold (Fig. 3.13).

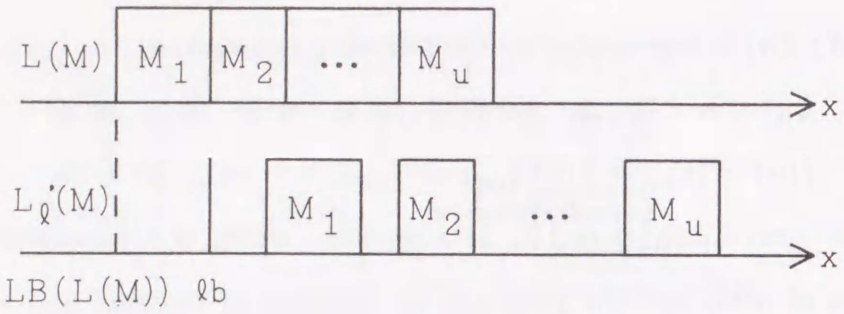


Figure 3.13: L_l'(M) in Lemma 3.3.

i) For the constant $\ell b \geq LB(P(M))$, the placement $L_l(M)$ in which $M_1, M_2, \cdots, M_u$ are concatenated, and which satisfies $LB(L_l(M)) = \ell b$, is the optimal among the placement $L_l'(M)$ which satisfies $LB(L_l'(M)) \geq \ell b$. And $Z(L_l(M))$ is non-increasing as ℓb.

ii) For the constant $rb \leq RB(L(M))$, the placement $L_r(M)$ in which $M_1, M_2, \cdots, M_u$ are concatenated, and which satisfies $RB(L_r(M)) = rb$, is the optimal among the placement $L_r'(M)$ which satisfies $RB(L_r'(M)) \leq rb$. And $Z(L_r(M))$ is nondecreasing as rb.

(Proof) i) Consider the placement $L_l'(M)$ of $M_1, M_2, \cdots, M_u$ in which there are spaces between modules, and which satisfies $LB(L_l'(M)) \geq \ell b$. We show that this placement can be transformed to the placement $L_l(M)$ in which all modules are concatenated and $LB(L_l(M)) = \ell b$, without increasing $Z(L_l'(M))$. Now, CMa is the concatenated module which is placed right to the right most space. From Lemma 3.2, Z(L(CMa)), which is the objective function of CMa in L(M), is nondecreasing or increasing when it

is placed right of LB(L(CMa)). Now, LB(L(CMa))≤LB(L_l' (CMa)), then Z(L_l'(M)) does not increase when CMa is moved to left and concatenated with the left module.

For all spaces, repeat this from right. Then L_l'(M) can be transformed to the placement in which modules are concatenated and which satisfies LB(L_l(M))=ℓb, without increasing the objective function.

It is obvious that the objective function is nondecreasing as ℓb. In the case of ii), it can be proved like i). □

[**Lemma 3.4**] For two subsets of modules Ma = $\{M_j, M_{j+1}, M_{j+2}, \cdots, M_{j+s-1}\}$, Mb = $\{M_{j+s}, M_{j+s+1}, M_{j+s+2}, \cdots, M_{j+s+\tau-1}\}$ (1 ≤ j ≤ j+s−1<j+s+τ−1 ≤ m) of a set of modules M = $\{M_1, M_2, \cdots, M_m\}$, assume that the optimal placements La(Ma) and Lb(Mb), which satisfy the condition C2, are given independently. If Ma and Mb are concatenated as CM< j, s > and CM< $j+s, t$ > in La(Ma) and Lb(Mb), there exists the placement L(Ma∪Mb) which consists of CM< $j, s+t$ > as a placement of Ma∪Mb(Fig. 3.14).

[**Condition C2**] LB(Lb(Mb))<RB(La(Ma)) □

(Proof) Assume that for M'=Ma∪Mb, the optimal placement P'(M'), which does not consist of CM< $j, s+t$ >, is given. We show that this can be transformed to the placement L(M') in which all modules are concatenated without increasing Z(L'(M')).

i) RB(L'(Ma))≤LB(Lb(Mb)): From the above assumption, Lb(Mb) is

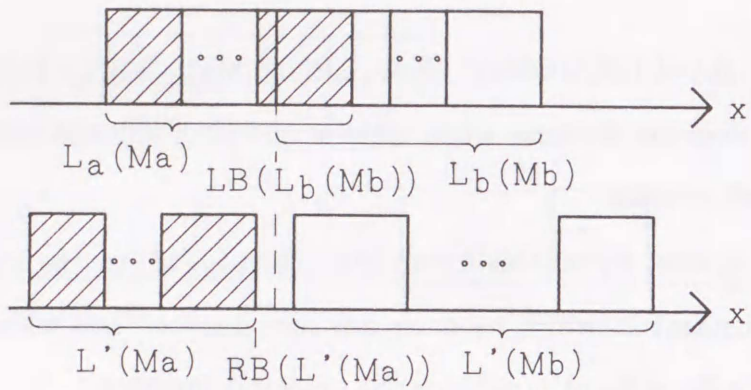


Figure 3.14: P'(M) in Lemma 3.4.

the optimal placement. Then L'(M') can be transformed to the placement in which L'(Mb) equals to Lb(Mb) without increasing Z(L'(Mb)). Assume that rb=LB(Lb(Mb)). Because rb$\leq$RB(La(Ma)) (from Condition C2), it can be transformed to the placement L(M') in which Ma is concatenated to Mb without increasing Z(L'(Ma)).

ii) RB(L'(Ma)) > LB(Lb(Mb)): From the above assumption, La(Ma) is the optimal placement. As i), L(Ma) is transformed to La(Ma). Assume that ℓb = RB(L'(Ma)). Because ℓb > LB(Lb(Mb)), it can be transformed to the placement L(M') in which Mb is concatenated to Ma as i).

[**Theorem 3.3**] Algorithm MPA obtains an optimal solution of Problem MPP. □

(Proof) Consider Ma=$\{M_1\}$. For a subset Ma $\subseteq$ M, the algorithm MPA obtains the solution so that Z(L(Ma)) is minimum (from Theorem 3.2) because Z(L(CM$<1,1>$))=Z(L(M_1)). Assume that Ma= $\{M_1, M_2, \cdots, M_{j-1}\}$, and

MPA can obtain the placement so that Z(L(Ma)) is minimum. Based on the above assumption, the optimal placement L(Ma) of Ma is given. And we show that when MPA places M_j, the placement with minimum Z(L(Ma $\cup\{M_j\}$)) is obtained, for Ma $\cup\{M_j\} = \{M_1, M_2, \cdots, M_j\}$. Algorithm MPA calls the procedure PLM for CM$< j, 1 >$.

1) When there exists a concatenated module which is already placed, and LB(L($CM < j, k >$))<RB(L($CM < r, u >$)) for $CM < r, u >$ which is placed right most:

In the optimal placement L(Ma), $CM < r, u >$, which is placed right most and consists of $M_r = \{M_r, M_{r+1}, \cdots, M_{j-1}\}$, is minimum with Z(L($\{M_j\}$)) From Theorem 3.2. Then there exists the placement which is minimum with Z($M_r \cup \{M_j\}$) from Lemma 3.4.

CM$< r, w >$ is obtained by concatenating these modules. For the placement L(CM$< r, w >$) with minimum Z(L(CM$< r, w >$)) is minimum with Z(L($M_r \cup \{M_j\}$)). Repeat this discussion until there is no overlaps. Then the placement with minimum Z(L(M)) is obtained.

2) Because Z(L($\{M_j\}$)) is minimum, Z(P(M)) is also minimum. □

3.3.4 Extension of Problem MPP

First, the following limitation is added to the problem MPP. The modules are allowed only in the *placement region*, $[x_l, x_r]$ which is given as input.

[Problem MPP1] The problem MPP1 is defined by modifying the problem MPP so that the placement region $[x_l, x_r]$ is given as its input. □

In the algorithm MPA1 for the problem MPP1, the procedure BP in the algorithm MPA is substituted by the following procedure BP1. In the procedure BP1, for the concatenated module $CM < j, k >$, the placement L($CM < j, k >$) is firstly obtained by the procedure BP. Regarding this placement,

i) if LB(L($CM < j, k >$)) $< x_l$, L($CM < j, k >$)=x_l,

ii) if $x_r <$ RB(L($CM < j, k >$)), L($CM < j, k >$)=x_r - $w(CM < j, k >)$,

iii) otherwise, output L($CM < j, k >$).

[**Theorem 3.4**] The algorithm MPA1 obtains the optimal solution of the problem MPP1. □

(Proof) From Lemma 3.2, it is trivial that the procedure BP1 obtains the optimum placement, which minimizes the Z(L({M_i})), for the module M_i. Then by the same discussion in the proof of Theorem 3.2, The algorithm MPA1 obtains the optimum solution of the problem MPP1. □

Because the procedure BP1 has the same time complexity of the procedure BP, the time complexity of the algorithm MPA1 is $O(|N| \cdot (|M| + log|N|))$, which is the same complexity of the algorithm MPA.

Next, each net is extended to the multi-terminal net. The problem is formulated as follows.

[**Problem MPP2**] For the problem MPP2, the logic circuit LC=(M,Te,N) in the problem MPP1 is extended as follows. M and Te are same as those in the problem MPP1. It is assumed that N represents the netlist of multi-

terminal nets which connect modules and external terminals. The problem MPP2 is defined as an extension of the algorithm MPP1, as mentioned above. □

[**Theorem 3.5**] The problem MPP2 is reducible to the problem MPP1 in polynomial time. □

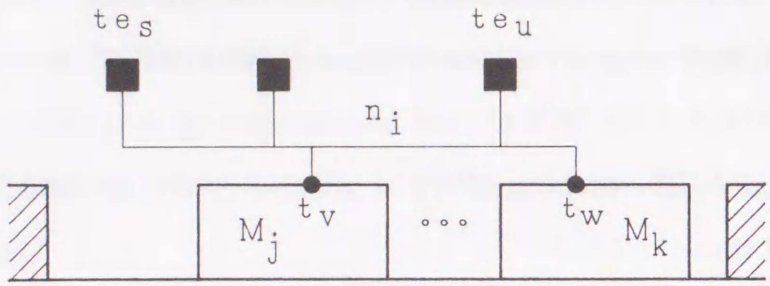
(Proof) In this proof, only the following transformations are given. The first translation is from the multi-terminal net n_i in the problem MPP2 to the four two-terminal nets including the net between modules. The second transformation is from the net between modules to the two nets, each of which connects the external terminal and the terminal of the module. Assume that a multi-terminal net n_i which connects external terminals and terminals of modules is given (see Fig. 3.15).

It is assumed that the left most external terminal and terminal of a module are te_s and t_v, respectively, and the right most external terminal and terminal of a module are te_u and t_w, respectively. Then four two-terminal nets $n_i^{(1)}$, $n_i^{(2)}$, $n_i^{(3)}$, and $n_i^{(4)}$ are constructed as follows (see Fig. 3.15 (b)).

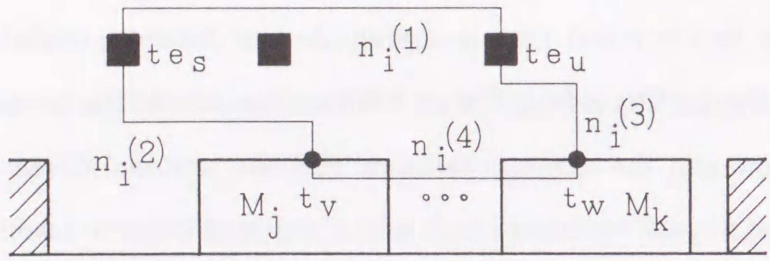
$n_i^{(1)} = te_s, te_u$, $n_i^{(2)} = te_s, t_v$, $n_i^{(3)} = te_u, t_w$, $n_i^{(4)} = t_v, t_w$.

Between the wire length $Zx(n_i)$ of the net n_i in the problem MPP2 and the wire lengths $Zx(n_i^{(1)}$, $Zx(n_i^{(2)}$, $\cdots$, $Zx(n_i^{(4)}$, the following equation holds.

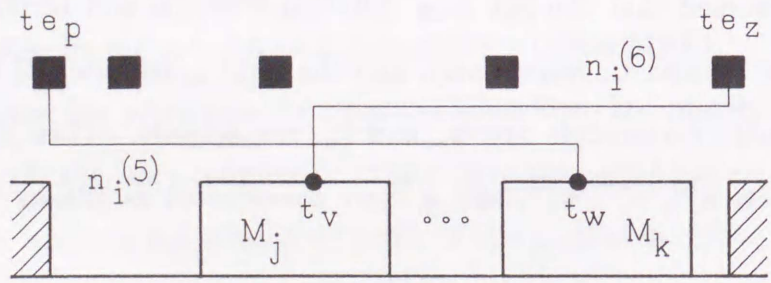
$$Zx(n_i^{(1)}) + Zx(n_i^{(2)}) + Zx(n_i^{(3)}) + Zx(n_i^{(4)}) = 2 \cdot Zx(n_i)$$



(a) Multi-terminal net



(b) Two-terminal nets including the net between modules



(c) Two-terminal nets

Figure 3.15: Translation of the net.

It is trivial that to minimize $Zx(n_i)$ equals to minimize $Zx(n_i^{(1)}) + Zx(n_i^{(2)}) + Zx(n_i^{(3)}) + Zx(n_i^{(4)})$.

Next, it is assumed that the net $n_i^{(4)}$, which connects terminals of two modules, is given. Then the two nets $n_i^{(5)}$ and $n_i^{(6)}$, each of which connect the external terminal and the terminal of the module, are constructed as follows (see Fig 3.15 (c)).

$n_i^{(5)} = t_w, te_p, n_i^{(6)} = t_v, te_z.$

Between the wire length $Zx(n_i^{(4)})$ of the net $n_i^{(4)}$ and the wire lengths $Zx(n_i^{(5)})$, $Zx(n_i^{(6)})$, the following equation holds.

$Zx(n_i^{(5)}) + Zx(n_i^{(6)}) = (D(te_z) - D(te_p)) + Zx(n_i^{(4)})$

It is trivial that to minimize $Zx(n_i^{(4)})$ equals to minimize $Zx(n_i^{(5)}) + Zx(n_i^{(6)})$.

These transformations can be executed in a constant time for each net in the problem MPP2, then the problem MPP2 can be reducible to the problem MPP1. □

The time complexity of the problem MPP2 is the same as that of the problem MPP1.

For the problem MPP1, the wire length minimization problem which includes the y coordinates as well as the x coordinates is considered. Assume that each net is a two-terminal net, which connects the external terminal and the terminal of the module, and that the river routing [0] is adopted as a routing model (see Fig. 3.16). This problem is formulated as the

following Problem MPP3.

[**Problem MPP3**] Problem MPP3 is defined as the problem that the objective function $L(n_i) = L_x(n_i) + y$ is substituted for $L_x(n_i)$ in Problem MPP1, where y is a distance between the modules and the external terminals (separation). □

[**Theorem 3.6**] There exists an algorithm which solves Problem MPP3 optimally in $O(|N|^2 \cdot (|M| + log|N|))$.

(Proof) According to the paper [0], the routing is possible for the given separation y if and only if $D(te_{i+y}) - D(t_i) \geq y$. As the external terminals are fixed, the above inequality can be transformed to the following ineqation on each terminal t_k for a given separation y in $O(|N|)$ time; $\alpha_k \leq t_k \leq \beta_k$. This inequality can be transformed again to the following inequality on each module M_j in $O(|N|)$ time; $\gamma_j \leq LB(P(M_j)) \leq \delta_j$.

Problem MPP1, in which the placement region for each module is given as input, can be solved optimally in $O(|N| \cdot (|M| + log|N|))$. The separation y takes the value between 0 and $|N|$. By executing the minimization of wire length on x coordinates for the all possible values of y, the placement with minimum wire length on the river routing model can be obtained in $O(|N|^2 \cdot (|M| + log|N|))$. □

Finally, the wire length minimization problem such that the initial placement is not given is considered. This problem is formulated as the following Problem MPP4.

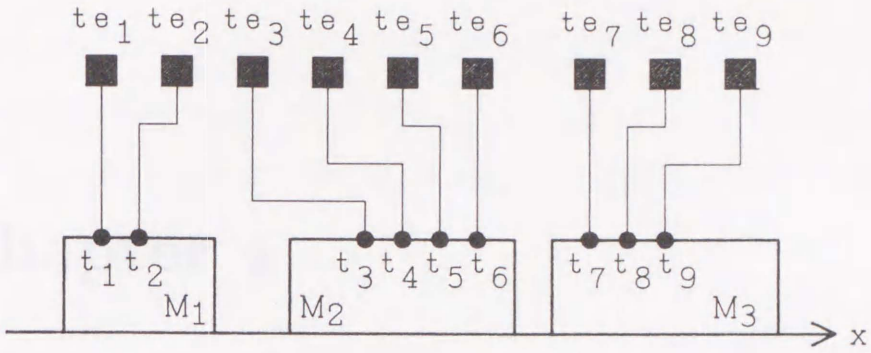


Figure 3.16: River routing.

[**Problem MPP4**] A logic circuit LC=(M,Te,N) and the placement region $[x_l, x_r]$ for modules is given as inputs.

Each net $n_i \in$ N consists of two terminals, one of which is a terminal t_j of a module and the another is an external terminal te_j. Then find the placement L(M) on x axis which satisfies the following condition i and condition ii, and minimizes the objective function $Z=\sum_{n_i \in N} c_i \cdot Zx(n_i)$, where c_i is the weight of the net n_i and $Zx(n_i)$ is the distance of two terminals on x axis.

(Condition i) Modules are not overlapped each other. (Condition ii) Modules are placed in the placement region.

[**Theorem 3.7**] Problem PM4 is NP-complete.

(Proof) See Appendix. □

Chapter 4

Hierarchical Floorplanning with Global Routing and Positioning

This chapter describes the heuristic algorithms of the hierarchical floorplanning with global routing and positioning. The proposed method consists of the following two stages: the initial floorplanning (Stage 1) and the detailed floorplanning (Stage 2). Section 4.1 describes the algorithms of the initial floorplanning, and Section 4.2 describes the algorithms of the detailed floorplanning.

4.1 Initial Floorplanning (Stage 1)

An *initial floorplan* consists of a rectangle which represents a chip, a placement L(Mh) of hard modules Mh and center points of virtual modules Mv in the rectangle. For a given logic circuit, the initial floorplanning determines such an initial floorplan. It consists of the following 4 phases:

the placement of module centers (Phase 1), the determination of module orientations (Phase 2), the resolution of overlaps (Phase 3), and the improvement of module placement (Phase 4). These four phases are explained in 4.1.1 - 4.1.4, respectively.

[Initial floorplanning]

Input: Logic circuit L

[**Phase 1**] Placement of module centers (Subsection 4.1.1);

[**Phase 2**] Determination of module orientations (Subsection 4.1.2);

[**Phase 3**] Resolution of overlaps (Subsection 4.1.3);

[**Phase 4**] Improvement of module placement (Subsection 4.1.4);

Output: Initial floorplan

4.1.1 Placement of module centers (Phase 1)

As inputs, a logic circuit L and the rectangle (called *chip region* R), which satisfies the given aspect ratio and has the area of $(1 + \gamma)$ times the total area of all modules, are given where γ is a constant. First, a set of virtual modules Mvg= $\{ Mv_j \}$ is constructed based on the group G such that each $Mv_j \in$ Mvg consists of a set of soft modules in the corresponding group G_j. Then the hard modules and virtual modules are regarded as points in

this phase. These points are placed by the algorithm based on the *ideal distance* described in Section 3.1.

[**Problem of Phase 1 in Stage 1**] Given a logic circuit L (all modules are regarded as points), the ID-matrix Dp=$[dp_{ij}]$, and the chip region A, obtain the positions of modules M= Mh ∪ Mvg in the chip region A so that the following objective function is minimized,

$$z = \sum_{for\ all\ ij} |dp_{ij} - dr_{ij}| \quad (i \neq j)$$

where dr_{ij} is an actual distance between M_i and M_j (M_i and $M_j \in M^* \cup P$). □

Using the procedure IDA in Section 3.1, the algorithm of Phase 1 in Stage 1 is given as follows.

[**Algorithm of Phase 1 in Stage 1**]

[**Step 1**] (Initialize) Construct a set of module Mvg based on the group G;

[**Step 2**] Place all modules in the chip region R by the random placement method [Watanabe 85]. (I/O pads $p_i \in P$ are assumed to have already been fixed around the chip region);

[**Step 3**] (Relative placement) Obtain a relative placement by calling the procedure IDA;

[**Step 4**] If the ratio of improvement is more than the given constant C, then go to Step 2; □

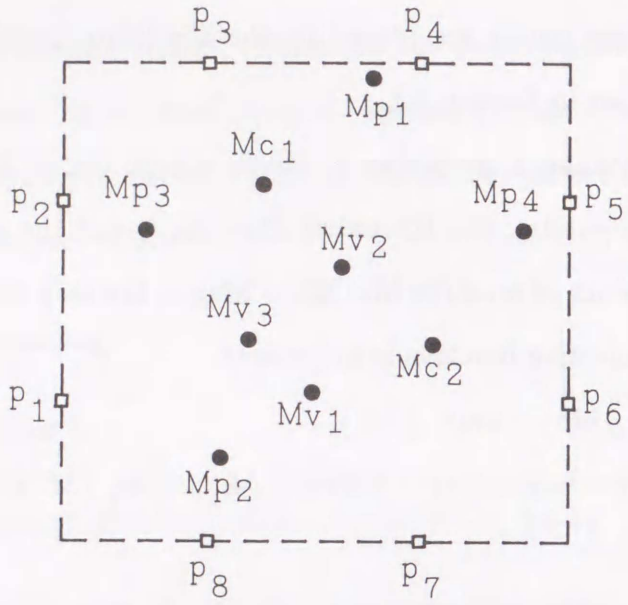


Figure 4.1: The result after Phase 1.

[**Example 4.1**] The result after Phase 1 is shown in Fig. 4.1. □

4.1.2 Determination of module orientations (Phase 2)

A set of hard modules Mh consists of a set of peripheral modules Mp and a set of center modules Mc. In Phase 2 and Phase 3 of Stage 1, the placement of center modules and virtual modules are determined, while the peripheral modules are ignored until Phase 4.

After the determination of positions of modules by the ideal distance method, actual shapes are given to the center modules, and orientations of them are determined. In order to reduce the computation time, a heuristic method is proposed which determines the orientation of each hard module

independently according to the wire lengths incident on it. While choosing the optimal orientation of each module from 8 possible types, other modules are regarded as points.

[**Problem of Phase 2 in Stage 1**] Given a set of hard modules Mh and the center positions of modules M in the chip region R, obtain the orientations for hard modules such that the total wire length is minimized. □

The algorithm of Phase 2 in Stage 1 is shown as follows.

[**Algorithm of Phase 2 in Stage 1**]

[**Step 1**] For all center modules $Mc_i \in$ Mc, execute the following Step 2 and Step 3;

[**Step 2**] Place a center module Mc_i so that the center of it is on the corresponding position obtained by Phase 1;

[**Step 3**] Choose the optimal orientation from 8 possible types so that the wire length for Mc_i is minimum;

[**Step 4**] Give the square shape to each virtual module with the area of α times the area for it ($\alpha < 1$), and place them so that the centers of them are located on the corresponding positions obtained by the algorithm of Phase 1; □

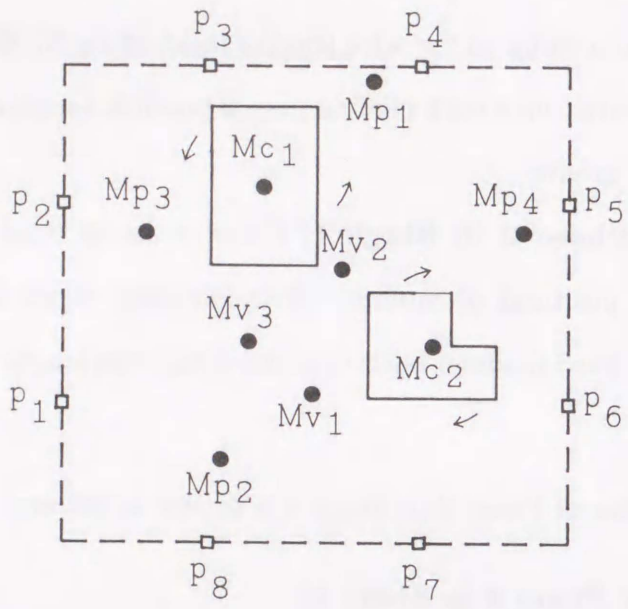


Figure 4.2: Possible orientations.

[**Example 4.2**] Possible orientations for Mc_1 and Mc_2 at Phase 2 are shown in Fig. 4.2. □

4.1.3 Resolution of overlaps (Phase 3)

Generally, there may exist overlaps among convex rectilinear shaped center modules Mc and small square shaped virtual modules in the result of Phase 2. These overlaps must be resolved. Furthermore, to avoid the unexpected increase of wire length for some nets, relative positions among these modules should be preserved during the overlap resolution. In addition to the problem defined in Section 3.3 which preserves relative positions among modules while resoluting overlaps, a given aspect ratio must be satisfied

in this problem of Phase 3. In the algorithm of Phase 3, the algorithm ORA is used as a procedure. In the algorithm ORA, pairs of modules in OL_x are resolved first by moving modules along x axis. The relative positions are not broken in this phase. While preserving relative position, pairs of modules in OL_y are resolved secondly by moving modules along y axis. Depending on the moving distance to resolve an overlap, the pair of overlapped modules belongs to OL_x or OL_y. The details are explained in Section 3.2.

By exchanging the elements of OL_x and OL_y, the aspect ratio of the obtained chip can be modified to satisfy the given aspect ratio. Then consider the sequence Sx of the subset of OL_x, and the sequence Sy of the subset of OL_y. In the following, the construction of the sequence Sx is explained.

First, the elements (M_i, M_j) of OL_x are sorted in the nondecreasing order of the value $|w_x(M_i, M_j) - w_y(M_i, M_j)|$. Then the | M | elements from the top of this sequence are chosen. Next, these | M | elements are sorted in the nonincreasing order of the value $w_x(M_i, M_j) + w_y(M_i, M_j)$. Then Sx is defined to be this sequence. Sy is defined in the same way. The i th element of Sx (or Sy) is denoted by sx_i (or sy_i).

In the algorithm of Phase 3, overlaps are resolved by the procedure ORA which consists of the phase 1 and the phase 2 of the algorithm ORA in Section 3.2. Then for all the modules which are already placed, the

aspect ratio (the height / the width) of the minimum enclosing rectangle is calculated. If this aspect ratio is less than the given aspect ratio, an element of OL_x is moved to OL_y according to the sequence Sx. And if the obtained aspect ratio is not improved, the element is moved back to OL_x.

If the obtained aspect ratio is greater than the given aspect ratio, the operation, in which OL_x and OL_y, and, Sx and Sy in the above explanation are exchanged, is executed. These operations are executed for all the elements of Sx or Sy.

[Problem of Phase 3 in Stage 1] Given a set of modules M (convex rectilinear polygon), $\mathrm{L_I(M)}$ in which overlaps of modules exist and, two positive constants c_1 and c_2, obtain $\mathrm{L_F(M)}$ which satisfies the following condition (i) and (ii), and minimize $z = c_1 \cdot \mathrm{D(L_I(M), L_F(M))} + c_2 \cdot \mathrm{A(L_F(M))}$.
(Condition i) $\mathrm{OL(L_F(M))} = \phi$
(Condition ii) $\mathrm{L_F(M)}$ preserves relative positions in $\mathrm{L_I(M)}$. □

The algorithm of Phase 3 in Stage 1 is shown as follows. The procedure ORA which is used in the algorithm is the phase 1 and the phase 2 of the algorithm ORA given in Section 3.2.

[Algorithm of Phase 3 in Stage 1]

(Phase 0 : Initialize)

[Step 1] For the initial placement $L_I(M)$, construct the two placement graph $G_x(L_I(M)) = (\bar{V}_{Ix}, \bar{E}_{Ix})$, and $G_y(L_I(M)) = (\bar{V}_{Iy}, \bar{E}_{Iy})$;

[**Step 2**] Obtain the sets OL_x, OL_y of module pairs which are overlapped each other; i $\leftarrow$ 1;

(Phase 1 : Initial resolution)

[**Step 3**] Remove overlaps among modules by the procedure ORA and store the value of the objective function to the variable: OBJECT;

[**Step 4**] Calculate the aspect ratio Rf of the minimum enclosing rectangle of all the modules;

[**Step 5**] If Rf $\leq$ R then go to Step 6; else go to Step 10;

/* Rf $\leq$ R */

[**Step 6**] If i $\leq$ | M | then go to Step 7; else go to Step 14;

[**Step 7**] LAST $\leftarrow$ OBJECT; remove sx_i from OL_x and put it in OL_y;

[**Step 8**] Remove overlaps among modules by the procedure ORA and store the value of the objective function to the variable: OBJECT; i $\leftarrow$ i+1;

[**Step 9**] If OBJECT < LAST then i = i+1, and go to Step 6; else remove sx_i from OL_y and put it back in OL_x;

/* Rf > R */

[**Step 10**] If i $\leq$ | M | then go to Step 11; else go to Step 14;

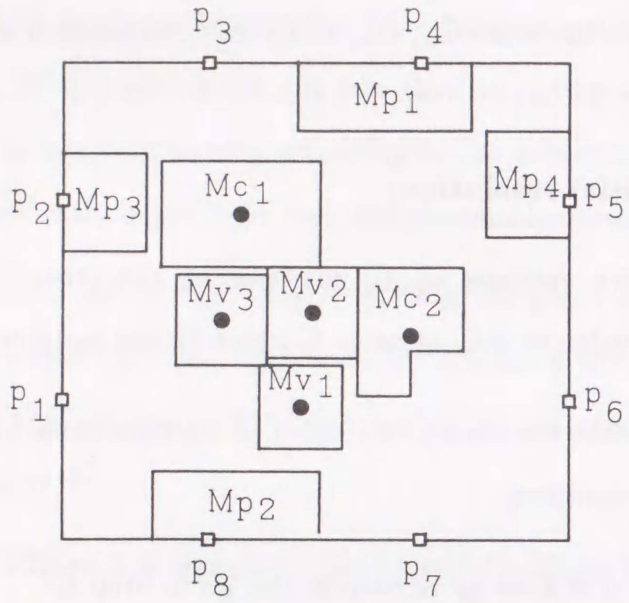


Figure 4.3: The result after Phase 3.

[Step 11] LAST $\leq$ OBJECT; remove sy_i from OL_y and put it in OL_x;

[Step 12] Remove overlaps among modules by the procedure ORA and store the value of the objective function to the variable: OBJECT; i $\leftarrow$ i+1;

[Step 13] If OBJECT < LAST then i $\leftarrow$ i+1, and go to Step 10; else remove sy_i from OL_x and put it back in OL_y; i $\leftarrow$ i+1; go to Step 10;

[Step 14] end. □

[Example 4.3] The result after Phase 3 is shown in Fig. 4.3. □

4.1.4 Improvement of module placement

Now consider the minimum enclosing convex rectilinear polygon for the placement of center modules and virtual modules. In this phase, these modules are regarded as one hypothetical module which has a rectilinear shape (see fig 4.5). In this phase, the initial placement of peripheral modules Mp is determined, and the placement of the hypothetical module and the peripheral modules are improved so that the total wire length is minimum.

First, the initial placement for peripheral modules is explained. Peripheral modules have strong connectivity with I/O pads and their orientation to the boundary of the chip is fixed. The total width of these modules is assumed to be less than the total length of the chip boundaries.

According to the center positions obtained in Phase 1, the peripheral modules are placed along the chip boundaries so that all peripheral modules have no overlap. The procedure IPP is described as follows.

[Procedure IPP]

[Step 1] Let the nearest peripheral module from the bottom left corner of the chip to be Mp'_1. Then let the j th peripheral module from Mp'_1 in counterclockwise around the center of the chip to be Mp'_j;

[Step 2] Divide the set of peripheral modules into four groups as follows. For the first group, the sum of the width of peripheral modules in the group is as wide as possible within $WA \times LB/LC$, where WA is

total width of the peripheral modules, LC is the total length of the chip boundaries, and LB is the width of the bottom boundary. The second, the third. and the forth groups are determined in the same way;

[**Step 3**] Place these peripheral modules along the corresponding boundaries with no overlap; □

Next the improvement of the placement of these peripheral modules and the hypothetical module are explained. By extending the algorithm MPA given in Section 3.3, all the placement of the peripheral modules on the four boundaries of the chip are improved optimaly by Procedure 4MPA, which is described as follows. The placement region which is on the bottom, right, top, or left boundaries of the chip is denoted by $[x_l, x_r]$ for each concatenated module.

[**Procedure 4MPA**]

[**Step 1**] For all $M_i \in$ M, execute the following Step 2 - Step 4;

[**Step 2**] $CM < i, 1 > = M_i$;

[**Step 3**] Construct $l_{<i,1>}$ for $CM < i, 1 >$;

[**Step 4**] Calculate P($CM < r, w >$) by calling the procedure 4PLM($CM < i, 1 >$, $[x_l, x_r]$) $(r + w - 1 = i)$; □

```
Procedure 4PLM(CM < j, k >, [x_l, x_r])
  begin
    P(CM < j, k >) ← 4BP(CM < j, k >, [x_l, x_r]);
    /* Let CM < r, u > be the right most
       concatenated module in PM. */
    if CM < j, k > is next to x_l then
      begin
        if x_l − RB(L(CM < r, u >)) < Ch and
            L(CM < r, u >) − x_l < Ch then
          begin
            Choose one according to the objective function;
            (i) PM ← PM − {CM < r, u >)};
              P(CM < r, u >) = 4PLM(CM < r, u >, [x_l, x_r − Ch])
              P(CM < j, k >) = P(CM < j, k >);
            (ii) P(CM < r, u >) = P(CM < r, u >);
              P(CM < j, k >) = 4BP(CM < j, k >, [x_l + Ch, x_r])
          end
        else
          begin
            PM ← PM ∪ {CM < j, k >};
            return P(CM < r, w >)
          end
      end
    else
      begin
        if LB(P(CM < j, k >)) < RB(P(CM < r, u >)) then
          begin
            PM ← PM − {CM < r, u >};
            Construct CM < r, w > (w=u+k) by concatenating
              CM < r, u > and CM < j, k >;
            Update l_<r,w> by merging l_<r,u> and l_<j,k>;
            P(CM < r, w >) = PLM(CM < r, w >, [x_l, x_r]);
            return P(CM < r, w >)
          end
```

```
        else
          begin
            PM ← PM ∪ {CM < j, k >};
            return P(CM < r, w >)
          end
      end
  end

Procedure 4BP(CM < j, k >, [x_l, x_r])
  begin
    /* Let l_<j,k> = (l_π(1), ···, l_π(r))
       be a terminal sorted sequence of CM < j, k > */
    Obtain the subscript π(q) such that
    c_π(1) + ··· + c_π(q) ≥ (c_π(1) + ··· + c_π(r)) / 2;
    if LB(L(CM < j, k >)) < x_l, then L(CM < j, k >)=x_l;
    if x_r < RB(L(CM < j, k >)), then
       L(CM < j, k >)=x_r - w(CM < j, k >);
    return L(CM < j, k >);
  end
```

Figure 4.4: Procedure 4PLM and 4BP.

Using the algorithm MPA, the placement of the hypothetical module Mth can be improved by Algorithm CA, which is described as follows.

[**Procedure CA**]

[**Step 1**] Improve the placement of the hypothetical module on x coordinates by calling the procedure MPA and obtain the new placement L_x(Mth);

[**Step 2**] Improve the placement of the hypothetical module on y coordinates by calling the procedure MPA and obtain the new placement L_y(Mth);

[**Step 3**] Output (L_x(Mth),L_y(Mth));

By calling the procedure IPP, 4MPA and CA, the placement of all hard modules is improved so that the total wire length is minimum.

[**Problem of Phase 4 in Stage 1**] Given a set of peripheral modules Mp and the center positions of them in the chip region R, obtain the placement of peripheral modules such that the total wire length is minimized. □

The algorithm of Phase 4 is described as follows.

[**Algorithm of Phase 4 in Stage 1**]

[**Step 1**] Obtain the initial placement of the peripheral modules by calling the procedure IPP;

[**Step 2**] $Z_{old} = \infty$, L(M) = L_I(M);

[**Step 3**] Improve the placement of the peripheral modules Mp by the procedure 4MPA.

[**Step 4**] Improve the placement of the hypothetical module Mth by the procedure CA.

[**Step 5**] Calculate Z_{new} for the obtained placement;

[**Step 6**] If ((Z_{old} - Z_{new}) / Z_{old}) $> \tau$ then $Z_{old} = Z_{new}$ and go to Step 2;

[**Step 7**] Output L(M);

[**Example 4.4**] The result after Phase 4 is shown in Fig. 4.5. □

4.2 Detailed floorplanning (Stage 2)

The detailed floorplanning consists of the detailed floorplanning at the level 0 and the detailed floorplanning at the level i ($1 \leq i \leq k$). The detailed floorplanning at the level 0 consists of the initial partitioning, the initial global routing, and the initial positioning. The detailed floorplanning at the level i consists of the routing-based partitioning, the hierarchical detailed global routing, and the hierarchical positioning. In Subsection 4.2.1, some preliminaries are explained for introducing the hierarchy in the detailed floorplanning. In Subsection 4.2.2, each algorithm in the detailed floorplanning at the level 0 is described, and the problem at the level i is

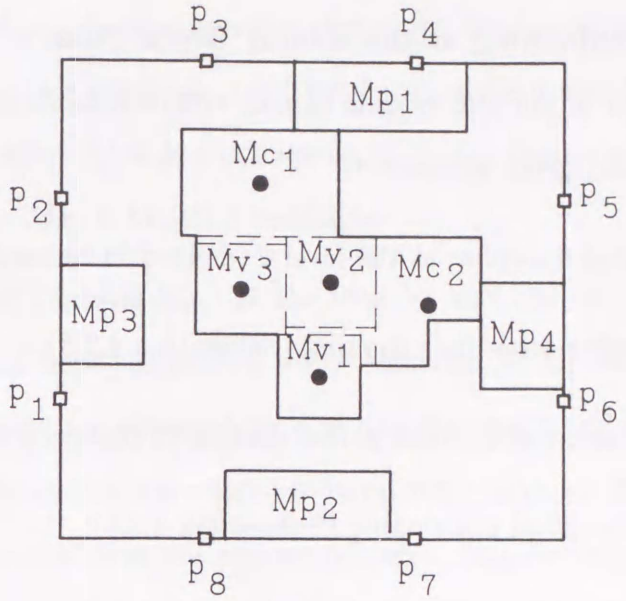


Figure 4.5: The result after Phase 4.

formulated. The algorithms of the detailed floorplanning at the level i are described in Subsection 4.2.3, 4.2.4, and 4.2.5, respectively.

[**Detailed floorplanning**]

Input An initial floorplan

(Detailed floorplanning at the level 0) (Subsection 4.2.1)

[**Phase 1**] Initial partitioning;

[**Phase 2**] Initial global routing;

[**Phase 3**] Initial positioning;

(Detailed floorplanning at the level i) Repeat Phase 4 - Phase 6 until the number of the soft module in each virtual module at the level i is less than the given constant C;

Input: A detailed floorplan at the level 0 (defined in Subsection 4.2.2)

[**Phase 4**] Routing-based partitioning (Subsection 4.2.3);

[**Phase 5**] Hierarchical detailed global routing (Subsection 4.2.4);

[**Phase 6**] Hierarchical positioning (Subsection 4.2.5);

Output: A detailed floorplan

4.2.1 Preliminaries on hierarchy

First, a set of modules at the level 0 and a chip region in the level 0 are defined.

[**Definition 4.1**] For a set of modules $M = Mh \cup Ms$ and a set of virtual modules Mg which are constructed in the initial floorplanning, the set of module M_0 at the level 0 and the chip region R_0 in the level 0 are defined respectively by the set of modules $Mh \cup Mv_0$ and the rectangle region which satisfy the following condition **A**.

(Condition **A**) (a) Mv_0 is a set of modules which are refinements of modules in Mg, where Mg is a set of virtual modules which are constructed in the initial floorplanning. (b) The rectangle is divided into several regions, and each divided region in the chip region has ono to one correspondence with

the module $M_j \in \mathrm{M}_0$, and has the area more than or equal to $(1 + \gamma) \cdot s(M_j)$, where γ is a constant. (c) The shape of the divided region for the hard module $Mh_j \in \mathrm{Mh}$ is a convex rectilinear polygon, and that for the virtual module $Mv_j \in \mathrm{Mv}_0$ is a rectangle. □

For a set of modules M_{i-1} at the level i-1 and the chip region R_{i-1} in the hierarchical floorplanning, a set of modules M_i is defined as the set of modules which are obtained by dividing the shape and the area of the virtual modules which have soft modules more than k. The chip region which is divided into several regions corresponding to the obtained M_i, is denoted by R_i. Each divided region in R_i for the module M_j is denoted by $R_i(M_j)$. The rectangular routing region which corresponds to each division line of R_i is called the *channel*, the cross region of the channels, which corresponds to the cross point of the division lines of R_i, is called the *switchbox*. Each channel can be divided into two routing region by the division line of R_i. Each divided routing region is called the *subchannel* (see Fig. 4.6).

R_i can be represented by the *channel graph* and the coordinates of the nodes of the graph. The *channel graph* $\mathrm{G}_i = (\mathrm{V}_i, \mathrm{E}_i)$ is an undirected graph such that

$\mathrm{V}_i = \{\ v \mid v$ is a node which corresponds to the switchbox $\}$

$\mathrm{E}_i = \{\ e \mid e$ is an edge which corresponds to the channel $\}$

For each edge $e \in \mathrm{E}_i$ in the channel graph $\mathrm{G}_i = (\mathrm{V}_i, \mathrm{E}_i)$, a weight $l(e)$ is

attached, which is the length of the channel of e. Assume that the channel e is adjacent to $R_i(M_j)$ and $R_i(M_k)$. For the corresponding subchannels of channel e, weights $k_s(e)$ and $pn_s(e)$ ($s = R_i(M_j)$, or $R_i(M_k)$) are attached. They are the capacity of the subchannel and the number of nets passing through the subchannel. For each channel e, $k(e)$ and $pn(e)$ are defined by the sum of those for the corresponding subchannels.

In this channel graph $G_i = (V_i, E_i)$, a subset of nodes which are on the boundaries of $R_i(Mv_j)$ the virtual module Mv_j are denoted by $Vb(Mv_j)$. **[Definition 4.2]** Given a channel graph $G_i = (V_i, E_i)$, and $Vb(Mv_j)$ for all virtual modules $Mv_j \in Mv_i$, *the global routing graph at the level i* is defined by the following weighted undirected graph $Gg_i = (Vg_i, Eg_i)$.

1. $Vg_i = \bigcup_{Mvj \in Mv_i} v(Mv_j) \cup V_i$, where $v(Mv_j)$ is a new node which corresponds to the region for $Mv_j \in Mv_i$.

2. $Eg_i = \bigcup_{Mvj \in Mv_i} E(Mv_j) \cup E_i$, where $E(Mv_j) = \{ (v(M_j), v) \mid v \in Vb(Mv_j) \}$.

3. The weight $w(e_k)$ for each edge $e_k \in Eg_i$ is

   $w(e_k) = C_4$ ($C_4 >> r(e_k)$) ($e_k \in E(Mv_j)$)

   $w(e_k) = r(e_k)$) (otherwise)

   where $r(e_k) = C_1 \cdot l(e_k) + C_2 \cdot pn(e_k) + C_3 \cdot (pn(e_k) - k(e_k))$,

   $C_1, C_2, \cdots, C_4$ are constants. □

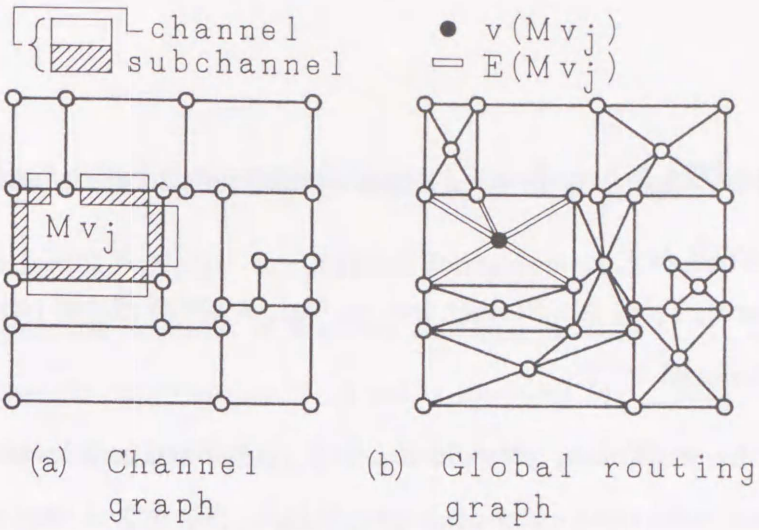


Figure 4.6: A channelgraph and its global routing graph.

[**Example 4.5**] An example of the channel graph $G_i = (V_i, E_i)$, and the corresponding global routing graph $Gg_i = (Vg_i, Eg_i)$ are shown in Fig. 4.6. □

For a virtual module $Mv_j \in Mv_i$, the terminals of the soft modules in Mv_j are represented by the hypothetical one terminal. For a set of modules $M_i = Mv_i \cup Mh$ and the netlist N in logic circuit L, the netlist N_i at the level i is defined by the all nets $n \in N$ which connect the same terminals of the modules of M_i and I/O pads. N_i is a set of all nets at the level i. The weight of the net $n \in N_i$ is t, if n corresponds to t nets in N.

For each net n at the level i, the *global route* is $T_i(n)$ on the global routing graph $Gg_i = (Vg_i, Eg_i)$ is defined by the tuple $T_i(n) = (V_i(n), E_i(n))$, where

$V_i(n) = \{ v \in Vg \mid v$ is a node which corresponds to the switchbox that n passes through $\}$

$E_i(n)$ = { $e \in Eg$ | e is an edge which corresponds to the channel that n passes through }

In general, $T_i(n)$ is a Steiner tree on Gg_i. A set of global routes at the level i is denoted by T_i.

Next, the *positioning graph at the level i* is defined as follows.

[**Definition 4.3**] Given a channel graph $G_i = (V_i, E_i)$ at the level i, the *vertical positioning graph* $Gpv_i=(Vpv_i, Epv_i)$ *at the level i* is a directed graph with weights, where

i) Vpv_i= { $v|v$ corresponds to the maximal subset of horizontal channels $e \in E_i$ in G_i }

ii) Epv_i= Ea ∪ Eb, where Ea and Eb are the sets of edges defined as follows. If there exist $R_i(M_j), \cdots, R_i(M_k)$ between horizontal channels on G_i, which correspond to the nodes $u, v \in Vpv_i$, corresponding edges $e_j, \cdots, e_k$ between u and v are defined. A set of these edges is denoted by Ea. If there does not exist such $R_i(M_j)$ and there exists a vertical channel e_s between horizontal channels which correspond to u and v, the corresponding edge e_s between u and v is defined. A set of these edges is denoted by Eb. □

[**Definition 4.4**] The horizontal positioning graph $Gph_i=(Vph_i,Eph_i)$ at the level i is defined as in Definition 4.1. □

4.2.2 Detailed floorplanning at the level 0 (Phase 1,2,3)

For a given initial floorplan, the detailed floorplanning at the level 0 determines the *detailed floorplan at the level 0*, which consists of (1) A channel graph G_0 and its coordinates, (2) A set of modules $M_0 = Mh \cup Mv_i$, (3) A placement of modules $L(M_0)$, (4) A netlist N_0, and (5) A set of global routes T_0. After initial floorplanning, hard modules $Mh_j \in Mh$ are already placed and a rectilinear region (it may have holes) may still remains, which contains the center positions of virtual modules $Mv_j \in Mvg$, where Mvg is a set of virtual modules constructed in the initial floorplanning based on the group G. To partition this rectilinear region into rectangles, *initial partitioning* is executed as follows. Rectilinear region for virtual modules in Mvg is divided into rectangles by extending the boundaries for hard modules $Mh_j \in Mh$. Then according to the ratio of areas of rectangles, each virtual module $Mv_j \in Mvg$ is partitioned. A set of obtained virtual modules is denoted by Mv_0. Then each virtual module $Mv_k \in Mv_0$ is assigned to the rectangle so that the virtual modules $Mv_k \in Mv_0$, which correspond to the original virtual module $Mv_j \in Mvg$, is placed around the center position of Mv_k, which is obtained by the initial floorplanning.

[Problem of Phase 1 in Stage 2] Given a set of modules $M = Mh \cup Mvg$ and their placement, obtain the set of modules M_0 at the level 0 and the chip region R_0 at the level 0 which satisfy the condition A in Definition 4.1. □

The algorithm of Phase 1 in Stage 2 is shown as follows.

[Algorithm of Phase 1 in Stage 2]

[Step 1] For all concave points of rectilinear polygon which consists of boundaries of $Mh_j \in \mathrm{Mh}$ and the chip in clockwise order around the center of the chip, execute the following Step 2 - Step 4.

[Step 2] Extend the boundary of Mh_j horizontally and vertically (the extended portion is called the segment), until it crosses another boundary or segment.

[Step 3] Delete the longer segment which is extended horizontally or vertically.

[Step 4] Partition virtual modules $Mv_j \in \mathrm{Mv}_g$ into virtual modules Mv_k and Mv_l by min-cut method [Fiduccia 82] according to the ratio of areas for all rectangles obtained in Step 2 (the obtained set of modules in Mv_0);

[Example 4.6] The result after Phase 1 is shown in Fig. 4.7. □

After the assignment of virtual modules $Mv_k \in \mathrm{Mv}_0$ to all rectangular areas, *initial global routing* is executed. For virtual modules in a rectangle, the positions of terminals are not fixed, and they are represented by one virtual terminal which is assumed to be located in the center of the rectangle.

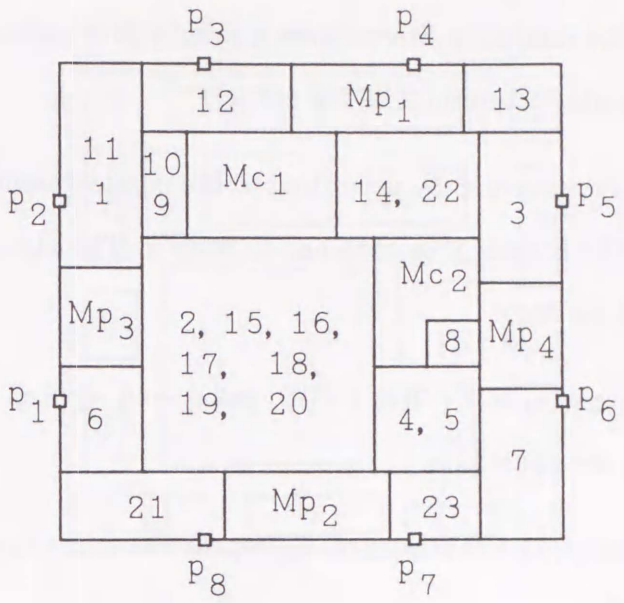


Figure 4.7: The result after Phase 1.

[**Problem of Phase 2 in Stage 2**] Given a netlist N and the global routing graph Gg_0, obtain the netlist N_0 at the level 0 and the set of global routes T_0 such that the total wire length of the global routes is minimized. □

The algorithm of Phase 2 in Stage 2 is shown as follows. In step 3, each Steiner tree is obtained by the algorithm in [3].

[**Algorithm of Phase 2 in Stage 2**]

[**Step 1**] For all edges e in $Gg_0 = (Vg_0, Eg_0)$, $w(e) = 0$;

[**Step 2**] $w(e) = l(e)$, and for each net $n \in N_0$, calculate $\alpha(n)$ which is the

length of the minimum Steiner tree whose Steiner points are $\{v(Mv_j)$ | Mv_j contains a terminal of the net n$\}$;

[**Step 3**] Sort the nets $n_l \in \mathrm{N}_0$ according to the nondecreasing order of the length of the Steiner tree obtained in Step 2. The obtained sequence is denoted by A_0;

[**Step 4**] Update w(e) to $C_1 \cdot l(e) + C_2 \cdots pn(e) + C_3 \cdot (pn(e) - k(e))$, where C_1, C_2, C_3 are constants.

[**Step 5**] According to the sequence A_0, repeat the following Step 6 for all nets $n \in \mathrm{N}_0$;

[**Step 6**] Obtain the minimum Steiner tree. And based on this tree, define the global route by $\mathrm{T}_0(\mathrm{n}) = (\mathrm{V}_0(\mathrm{n}), \mathrm{E}_0(\mathrm{n}))$. For each $\mathrm{e} \in \mathrm{E}_0(\mathrm{n})$, pn(e) = pn(e) + 1;

After the initial global routing, the width of each channel can be estimated. Then, the shape and the precise position of each module which minimizes the chip area is calculated (the initial positioning).

[**Problem of Phase 3 in Stage 2**] Given a channel graph G_0 at the level 0, their coordinates, and the global routes T_0, Obtain the placement $\mathrm{L}(\mathrm{M}_0)$ and updated coordinates of G_0. □

The initial positioning algorithm of Phase 3 in Stage 2 is the same as the algorithm of the hierarchical positioning. This algorithm is given in Section 4.2.5.

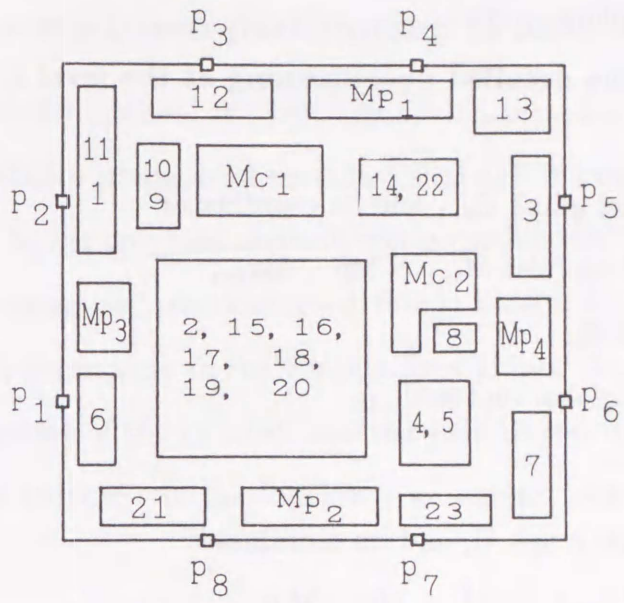


Figure 4.8: The result after Phase 3.

After the detailed floorplanning at the level 0, the *routing-based partitioning*, the *hierarchical detailed global routing*, and the *hierarchical positioning* (the detailed floorplanning at the level i) are executed repeatedly until the number of modules contained in each divided rectangle will be less than a given constant C. These three phases are explained in the following subsections.

[**Example 4.7**] The result after Phase 3 is shown in Fig. 4.8. □

After the Phase 3 in Stage 2, the channel graph G_{i-1} and its coordinates, a set of modules M_0, the netlist N_{i-1}, and the global routes T_{i-1} are determined. The problem of the detailed floorplanning at the level i is

formulated as follows.

[Problem of the detailed floorplanning at the level i]

Input:

(1) A channel graph G_{i-1} and its coordinates

(2) A set of modules $M_{i-1} = Mh \cup Mv_{i-1}$

(3) A netlist N_{i-1}

(4) A set of global routes T_{i-1}

Output:

(1) A channel graph G_i and its coordinates

(2) A set of modules $M_i = Mh \cup Mv_i$

(3) A placement of modules $L(M_i)$

(4) A netlist N_i

(5) A set of global routes T_i

Objective function:

Z = (the area of R_i) □

The algorithm for this problem consists of three phases: Phase 4, Phase 5, and Phase 6. These phases are complicated and it is difficult to formulate the problem for each phase, independently. In the following subsections, each phase is described.

4.2.3 Routing-based partitioning (Phase 4)

In the conventional method, the topology of channel graph is determined only by the relative positions of modules, although it greatly affects the global routes. In the proposed method, which combines floorplanning, detailed global routing, and positioning together in a hierarchical fashion, new channels which correspond to the division lines of modules are created so that the congestion of the channels and the switchboxes in the next level of hierarchy is reduced. In the following, we explain how to create new channels.

For each rectangle $R_{i-1}(Mv_j)$ for virtual modules $Mv_j \in \mathrm{Mv}_{i-1}$ in the floorplan in leve i-1, let the sets of edges which correspond to top (or bottom, left, right) side of $R_{i-1}(Mv_j)$ be Et (or Eb, El, Er). For the weight $pn_s(e)$ of each subchannel, $\mathrm{nt} = \mathrm{c} \cdot \lambda \cdot \max_{e \in Et} \{ pn_s(e) \}$, where λ is a width of the wiring grid and c is a constant. nb, nl, and nr are defined in the same way as nt. For Et in G_{i-1}, the set of corresponding edges in G_i is represented by Et'. A new channel which divides $R_i(Mv_j)$ horizontally (vertically) is denoted by e_h (e_v). The created rectangles are represented by $R_i(Mv_a)$ and $R_i(Mv_b)$, and the former is assumed to be placed above or left to the latter. The area of $R_i(Mv_a)$ is denoted by $\mathrm{S}(R_i(Mv_a))$.

[Procedure CUT]

[Step 1] For given $R_i(M_j)$, if M_j is a hard module or a virtual module

which contains less than or equal to k soft modules, then go to Step 6;

[**Step 2**] If nt+nb is greater than nl+nr, divide $R_i(Mv_j)$ so that

S($R_i(Mv_j)$) / S($R_i(Mv_j)$) = nb / nt;

else go to Step 4;

[**Step 3**] For each subchannel of $R_i(Mv_a)$ (represented by s), $k_s(e)$ = nt/2 ($e = e_h$ or $e \in$ Et'), $k_s(e)$ = nl ($e \in$ El'), $k_s(e)$ = nr ($e \in$ Er'). For each subchannel of $R_i(Mv_b)$, determine the weights in the same way. Then update each $l(e)$;

[**Step 4**] Divide $R_i(Mv_j)$ so that

S($R_i(Mv_j)$) / S($R_i(Mv_j)$) = nr / nl;

[**Step 5**] For each subchannel of $R_i(Mv_a)$ (represented by u), $k_u(e)$ = nl/2 ($e = e_v$ or $e \in$ El'), $k_u(e)$ = nt ($e \in$ Et'), $k_u(e)$ = nb ($e \in$ Eb'). For each subchannel of $R_i(Mv_b)$, determine the weights in the same way. Then update each $l(e)$;

[**Step 6**] For $R_{i-1}(M_j)$ (represented by w), $k_w(e)$ = nz ($e \in$ Ez'). Then update each $l(e)$;

In the following, the procedure PART, which partitions the virtual modules $Mv_j \in$ in $R_{i-1}(Mv_j)$ into two virtual modules Mv_a and Mv_b, is explained.

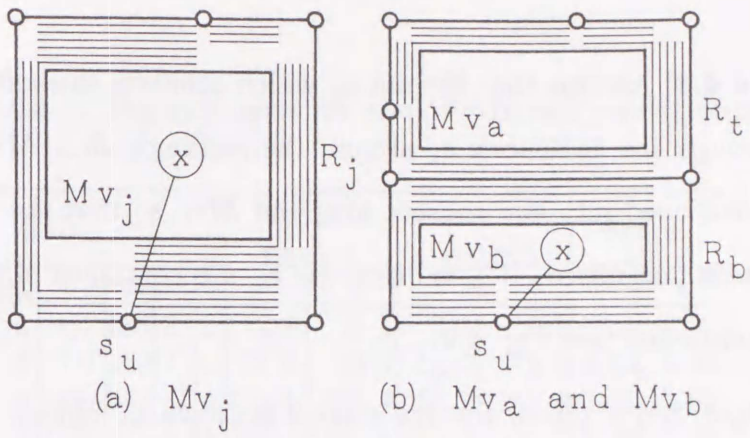


Figure 4.9: Routing-based partitioning.

[Procedure PART]

[Step 1] For the net n_k at the level i-1 which connects the virtual module Mv_j, find the passing through switchbox s_k on the periphery of $R_{i-1}(Mv_j)$;

[Step 2] On this switchbox s_k, put the hypothetical terminal of the net n_k;

[Step 3] Partition virtual module $Mv_j \in$ in $R_{i-1}(Mv_j)$ into two virtual modules Mv_a and Mv_b according to the ratio of S($R_i(Mv_a)$ and S($R_i(Mv_b)$ by the min-cut method [Fiduccia 82];

[Step 4] Divide the shape of each virtual module $Mv_j \in \mathrm{Mv}_{i-1}$ (rectangle) in the same direction of the division line of $R_i(Mv_j)$ (the shape of Mv_a and Mv_b are obtained);

[**Example 4.8**] Assume that the net n_k which connects the soft module x passes through the switchbox s_k around the rectangle $R_{i-1}(Mv_j)$. Then, Mv_j is partitioned into two subsets Mv_a and Mv_b, so that the number of the cut line is minimized. If s_k is below the e_h, x is contained in Mv_b, when Mv_j is partitioned (see Fig. 4.9). □

The algorithm of phase 4 in the stage 2 is shown as follows.

[**Input**] (1) a channel graph G_{i-1} and its coordinates (2) a set of modules M_{i-1} (3) a set of global routes T_{i-1}

[**Output**] (1) a channel graph G_i (weights $l(e)$ and k_s(e), except $pn_s(e)$) (2) a set of modules M_i

[**Algorithm of Phase 5 in Stage 2**]

[**Step 1**] For each rectangle R_j in which the virtual module Mv_j at the level i-1 is placed, call the procedure DIV and obtain the two subdivided rectangles R_t (R_l) and R_b (R_r).

[**Step 2**] Partition soft modules which constitute the virtual modules Mv_j into two new virtual modules Mv_t (Mv_l) and Mv_b (Mv_r), by calling the procedure PART.

We call this partitioning method of the virtual modules the *routing-based partitioning*.

The simulation experiments are done to compute the routing-based partitioning with the partitioning method which is based on only the number

Table 4.1: Experiments on routing-based partitioning.

Input data					Proposed method		Min-cut method		Ratio	
No.	n	ms	mh	h	ℓ	s	ℓ	s	w	a
1	191	250	2	5	2.9	10.3	3.1	10.4	7.54	-0.82
2	319	250	2	5	11.6	40.9	11.7	42.5	0.31	3.80
3	1249	493	3	6	103.1	363.4	103.3	369.4	0.21	1.63
4	1296	350	2	4	307.1	1002.4	321.2	1101.3	4.39	8.98
5	1258	479	2	5	353.6	1175.5	359.0	1209.3	1.49	2.80
6	1296	481	2	5	407.6	1316.2	416.9	1235.6	2.23	-6.52
7	1208	489	2	5	346.7	1144.5	355.1	1182.3	2.37	3.20

n:number of nets h:maximum level of hierarchy
ms:number of soft modules ℓ ($\times 10^4 \lambda$) :wire length
mh:number of hard modules s ($\times 10^4 \lambda^2$) :chip area
w (%) :improvement on ℓ a (%) :improvement on s

of the cuts [Fiduccia 82]. The input data and the results are shown in Table 4.1. The results show that the area of the floorplan obtained by the proposed method is smaller by 2.1% (ave.), and the total length of the global routing obtained by our method is also smaller by 2.6% (ave.). The effectiveness of the proposed method is verified by this experiments.

4.2.4 Hierarchical detailed global routing (Phase 5)

In our method, the global routes which directly correspond to the channels and the switchboxes are determined simultaneously with floorplanning. These routes are called the *detailed global routes* in order to distinguish the rough global routes for floorplanning used in the layout system [Dai 86].

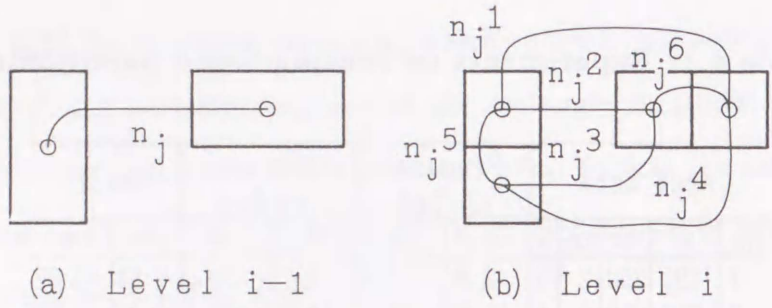


Figure 4.10: Division of the net.

The hierarchical detailed global routing is executed repeatedly with the routing-based partitioning and the hierarchical positioning. In order to execute the global routing hierarchically, the hierarchy is given to the nets in our method.

Assume that $\mathbf{N}_{i-1} = \{ n_1, n_2, \cdots, n_p \}$ and the net $n_j \in \mathbf{N}_{i-1}$ connects some virtual modules at the level i-1. Then at the level i, each virtual module is partitioned into two new virtual modules, and the net n_j is also divided into a subset $\mathbf{N}_i^j = \{ n_j^1, n_j^2, \cdots, n_j^q \} \subseteq \mathbf{N}_i$. Then the netlist $\mathbf{N}_i$ is represented by $\mathbf{N}_i^1 \cup \mathbf{N}_i^2 \cup \cdots \cup \mathbf{N}_i^p$.

[**Example 4.9**] An example of the net division is shown in Fig. 4.10. In this example, the net n_j at the level i-1 is divided into 6 nets n_j^1, n_j^2, $\cdots$, n_j^6 at the level i. □

In order to refer the global route determined in the previous level of hierarchy, each global route for the net at the level i is searched in a subgraph of the global routing graph Gg_i, which is constructed around the

global route of the corresponding net at the level i-1 (called the *reference line*). This subgraph is referred to as the *route search region*. By introducing the route search region, the computation time of the global routing can be reduced.

In general, there exist several optimal global routes for one net. In the proposed method, one of the optimal route is selected, and based on this route the division lines of modules is determined. If another optimal route is selected, a different result of module partition might be obtained. By restricting the route search region around the reference line, one optimal route is succeeded to the next level of hierarchy, and the correspondence between the global routes and the floorplan holds.

The minimum chip size may be obtained if the route search region is moderately restricted around the route which was obtained in the previous level of hierarchy. In order to verify this effect, simulation experiments are executed as follows. The reduction of the computation time is also expected.

Table 2 shows the experimental results of the total chip area and computation time where the types of the route search region are changed. In the experiment, route search region is restricted by the following Type 1 - Type 4, and compared by Type 5 which is not restricted.

- Type 1: channels around the modules which are adjacent to the reference line.

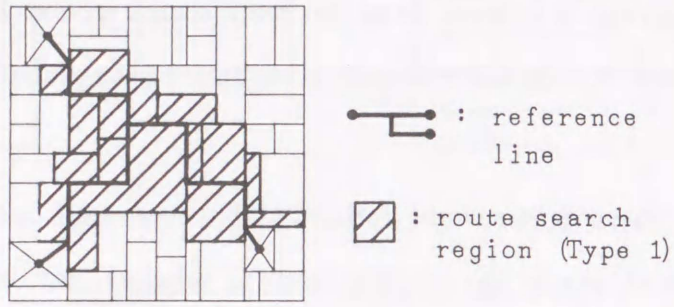


Figure 4.11: The route search region.

- Type 2: in addition to type 1, channels around the modules which are adjacent to type 1.
- Type 3: in addition to type 2, channels around the modules which are adjacent to type 2.
- Type 4: in addition to type 3, channels around the modules which are adjacent to type 3.
- Type 5: all channels in the global routing graph.

[**Example 4.10**] An example of a reference line and the corresponding route search region (Type 1) are shown in Fig. 4.11. □

The experimental results of the route search region is shown in Table 4.2. The results show that the total chip area is maximally reduced by 2.16% (ave.) when the route search region is set to Type 3. The computation

Table 4.2: Experiments on route search region.

Type	Chip area ($\times 10^4 \lambda^2$)				CPU time (sec.)			
	Data No.			Ave.*	Data No.			Ave.*
	1	2	3		1	2	3	
Type1	3. 71	8. 20	88. 1	+3. 75%	96	150	716	−59. 8%
Type2	3. 76	7. 89	82. 7	+0. 77%	128	190	1042	−46. 3%
Type3	3. 47	7. 73	83. 7	−2. 16%	160	240	1308	−32. 5%
Type4	3. 53	8. 00	82. 8	−0. 81%	184	273	1636	−21. 0%
Type5	3. 65	7. 95	82. 8	——	209	316	2635	——

Ave.*: (Type i − Type 5) / Type 5

time is reduced by 32.5% (ave.) in this case. Thus, it is concluded that setting the route search region to Type 3 makes it possible to obtain the minimum chip area as well as to reduce the computation time.

Next, the algorithm of Phase 5 in Stage 2 and some definitions for the algorithm are explained as follows. Assume that the sequence which is obtained by sorting the nets $n \in \mathrm{N}_i$ according to the nondecreasing order of the length of the Steiner tree is denoted by A_i, and that the j th net $n_{\pi(j)} \in \mathrm{N}_{i-1}$ in the sequence A_{i-1} at the level i-1 is simply denoted by n_j. Then for a subset of nets at the level i which is obtained by dividing n_j at the level i-1, the sequence which is obtained by sorting these nets according to the nondecreasing order of the length of the Steiner tree is denoted by $\mathrm{B}_i^j = (n_j^1, n_j^2, \cdots, n_j^k)$.

[**Input**] (1) a channel graph G_i (weights $l(e)$ and k_s(e), except $pn_s(e)$) (2)

a netlist N_{i-1} (3) a set of global routes T_{i-1}

[Output] (1) a channel graph G_i (weight $pn_s(e)$) (2) a netlist N_i (3) a set of global routes T_i

[Algorithm of Phase 5 in Stage 2]

[Step 1] Obtain the sequence A_{i-1} from N_{i-1};

[Step 2] Construct N_i by dividing $n_j \in N_{i-1}$;

[Step 3] From N_i, construct B_i^j for each $n_j \in N_{i-1}$;

[Step 4] Construct Gg_i from G_i and let w(e) be $C_1 \cdot l(e) + C_2 \cdots pn(e)$ $+C_3 \cdot (pn(e) - k(e))$, where C_1, C_2, C_3 are constants;

[Step 5] According to the sequence A_{i-1}, repeat the following Step 6-8 for all nets $n_j \in N_{i-1}$;

[Step 6] By referencing the net $n_j \in N_{i-1}$, construct the route search region of the type 3 on the global routing graph Gg_i;

[Step 7] According to the sequence B_i^j, repeat the following Step 8 for each net n in B_i^j.

[Step 8] Obtain the minimum Steiner tree, and based on this tree, define the global route by $T_i(n) = (V_i(n), E_i(n))$; For each $e \in E_i(n)$, $pn(e)$ $= pn(e) + 1$;

[**Step 9**] Divide each $pn(e)$ to the corresponding subchannels $pn_u(e)$ and $pn_w(e)$ according to the ratio of $k_u(e)$ and $k_w(e)$, where u and w represent $R_i(Mv_j)$ and $R_i(Mv_k)$ for e, respectively.

4.2.5 Hierarchical positioning (Phase 6)

After the hierarchical detailed global routing of level i, each virtual module $Mv_j \in \mathrm{Mv}_i$ is reshaped and the coordinates of G_i are updated in order to reduce the dead space. First, the procedure RE, which reshapes the virtual modules, is explained. The width and the height of Mv_J are represented by wm and hm, respectively. Consider the region (generally, it is a rectilinear polygon) that $\lambda \cdot pn_s(e)$ (s= $R_i(Mv_j)$) is deleted from each boundary of $R_i(Mv_j)$. The maximum rectangle in this region is referred to as the *placeable regin* of Mv_j. The width and the height of the placeable region is denoted by wp and hp, respectively. Whenever the procedure is called, wm $\leq$ wp and hm $\leq$ hp hold.

[Procedure RE]

1. Case wp = wm and hp = hm: As Mv_j corresponds to the edge on the longest path on Gpv_i and Gph_i, Mv_j is no reshaped;

2. Case wp < wm and hp = hm: As Mv_j corresponds to the edge on the longest path on Gpv_i, Mv_j is reshaped such that wp = wm and hp < hm;

3. Case wp = wm and hp < hm: As Mv_j corresponds to the edge on the longest path on Gph_i, Mv_j is reshaped such that wp < wm and hp = hm; □

The algorithm of Phase 6 in Stage 2 is shown as follows.

[**Input**] (1) a channel graph G_i (2) a set of modules M_i

[**Output**] (1) a channel graph G_i (weights $l(e)$) and its coordinates (2) a netlist N_i (3) a placement $L(M_i)$

[**Algorithm of Phase 6 in Stage 2**]

[**Step 1**] Construct the positioning graphs $Gpv_i = (Vpv_i, Epv_i)$, $Gph_i = (Vph_i, Eph_i)$;

[**Step 2**] Obtain the placement of modules $L(M_i)$ by the compaction method [Asano 86];

[**Step 3**] Repeat Step 4 - 6 until the chip size is not changed;

[**Step 4**] Obtain the subset of modules $Mb \subseteq Mv_i$, which corresponds to the edge Eb;

[**Step 5**] Reshape $Mv_j \in Mb$ by the procedure RE;

[**Step 3**] Obtain the coordinates of G_i, the weight $l(e)$, and the placement $L(M_i)$ by the compaction method [Asano 86];

[**Step 7**] Calculate the capacity $k(e)$ of each channel e;

Table 4.3: Experiments on hierarchical positioning.

Input data					Proposed positioning		Conventional positioning		Ratio	
No.	n	ms	mh	h	ℓ	s	ℓ	s	w	a
1	120	203	2	3	44.6	109.8	53.6	127.7	16.7	14.1
2	468	100	2	4	126.1	637.7	128.7	686.2	2.0	7.1
3	468	200	2	5	179.8	749.2	203.4	749.2	11.6	0.0
4	520	156	2	4	186.6	771.1	193.4	776.7	3.5	0.7
5	600	161	2	5	200.5	794.1	202.7	841.6	1.1	5.6
6	771	197	2	5	362.2	1166.3	410.3	1216.7	11.7	4.2
7	1150	299	2	5	687.2	1677.0	703.6	1667.1	2.3	−0.6

This hierarchical positioning is compared by the positioning which is applied only once as the final phase of the design. The input data and the results are shown in Table 4.3. The results show that the area of the floorplan obtained by our method is smaller by 4.4% (ave.), and the total length of the global routing obtained by our method is also smaller by 7.0% (ave.). The effectiveness of our method is verified by this experiments.

[**Example 4.11**] The final result for the input of Example 2.7 is shown in Fig. 4.12. □

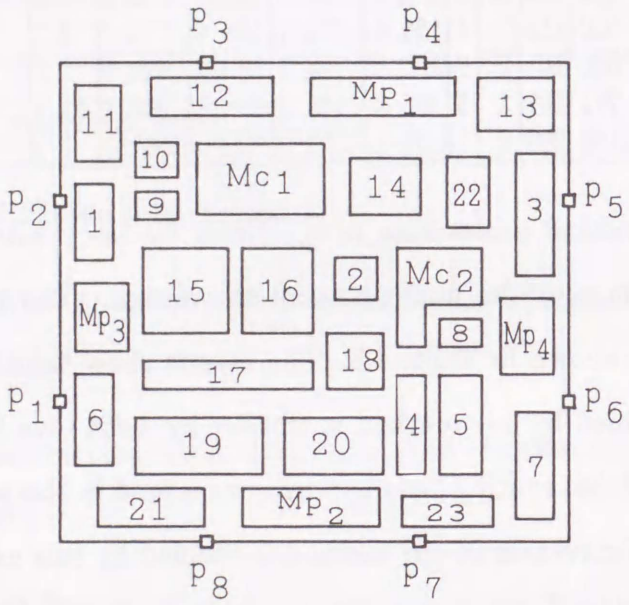


Figure 4.12: Final result.

Chapter 5

Evaluation of New Floorplanning Method

5.1 Hierarchical Floorplanning System FLORA II

Based on the hierarchical floorplanning method which combines global routing and positioning presented in this dissertation, a prototype floorplanning system FLORA II was implemented in the C language on an AV310 workstation (20 MIPS)

To evaluate this system, FLORA II has been compared with the conventional method which consists of a Min-cut method [Fiduccia 82] and a global router [Kou 78]. The characteristics of seven test data (Data No.1 - No.7) are summarized in Table 5.1 (Data No.8 is explained in the next section), where n is the number of the nets, ms is the number of the soft modules, mh is the number of the hard modules, and h is the highest level of the hierarchy. The results are also summarized in Table 5.1, where l is

the total wire length, s is the chip area. CPU time (sec.) for each data is also listed in the table.

The results show that the wire length is reduced by 14.7% (ave.), and at the same time, the area of the chip is reduced by 4.1% (ave.), respectively. The area of the chip is optimized including the routing area in FLORA II, whereas the area of the chip is increased during the global routing in the conventional method because the conventional floorplanner cannot predict the presice routability. The floorplans obtained by the proposed method and the conventional method for the data No.4 are shown in Fig. 5.1 and Fig. 5.2, respectively.

Compared to the conventional method, FLORA II takes about three times longer CPU tome. But even for Data No.7, which is the largest data, it takes only 52 minute., which is considered to be practical.

These results lead to the conclusion that the proposed method can obtain a better floorplan than the one produced by the conventional method.

There is another merit of the new approach. The proposed method gives a set of detailed global routes to a given net list, and thus detailed routers (channel routers, etc.) can easily be applied to the result of the floorplanner to perform detailed routing.

Table 5.1: Experimental results.

Input data					FLORA II			Conventional method		
No.	n	ms	mh	h	ℓ	s	sec.	ℓ	s	sec.
1	120	203	2	3	44.6	109.8	46	59.1	115.9	32
2	468	100	2	4	126.1	637.7	312	155.5	704.6	113
3	468	200	2	5	179.8	749.2	804	234.5	752.9	224
4	520	156	2	4	186.6	771.1	436	195.3	784.5	155
5	600	161	2	5	200.5	794.1	920	249.0	844.4	207
6	771	197	2	5	362.2	1166.3	1417	409.0	1219.4	397
7	1150	299	2	5	687.2	1677.0	3144	687.8	1703.4	1028
8	120	203	2	3	62.0	137.2	—	75.6	143.1	—

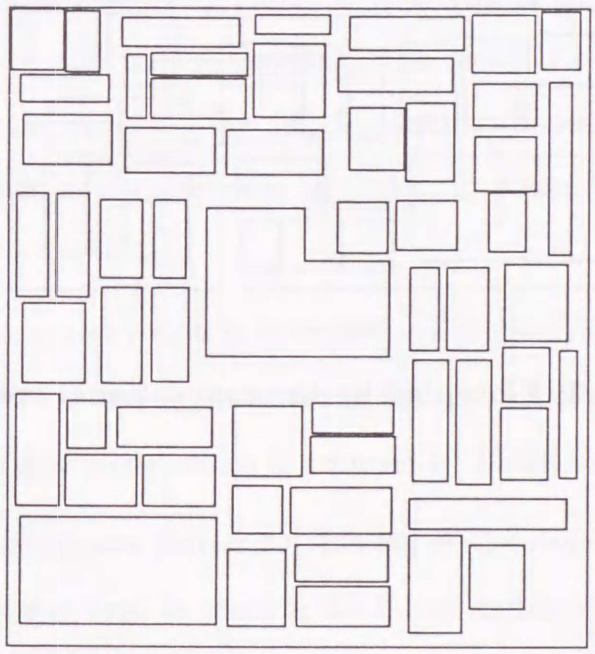


Fig. 5.1: Floorplan by the proposed method.

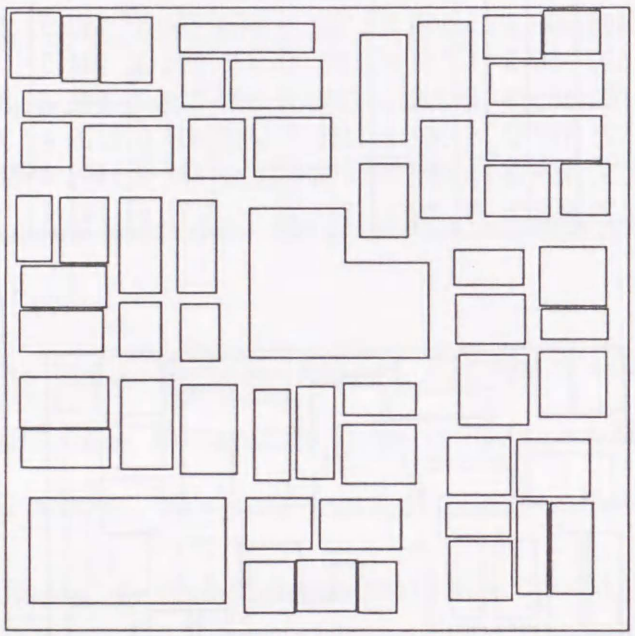


Fig. 5.2: Floorplan by the conventional method.

5.2 Extension to the total layout system

The prototype floorplanning system FLORA II was extended to the total layout system, referred to as FLORA II$^+$, by adding the terminal positioner [Ohmura 90], the channel router [Chen 86], and the switchbox router [Rivest 82] [Lin 89].

In the proceeding [Ohmura 90a], the terminal positioning problem is formulated as the minimization problem on the total wire length of nets between switchboxes and terminals under certain conditions. This problem is optimally solved by applying the algorithm discussed in Section 3.3 with some modifications. The positions of terminals affect the width of each routing area, and in this algorithm, the width of each routing area may sometimes increase, and the dead spaces may occur. One of the future research is to develop another algorithm to determine the terminal positions.

The original channel router in [Chen 86] is a gridless router. This channel router is modified and implemented so that the detailed routing can be executed on the grid model which is assumed by FLORA II$^+$.

Two swichbox routers [Rivest 82] [Lin 89] are implemented in FLORA II$^+$. The first one is used to execute the initial switchbox routing. Then several nets are ripped up and rerouted by the switchbox router [Lin 89].

Figures 5.3 and 5.4 show the final layout by the FLORA II$^+$ and the

conventional method mentioned in Section 5.1 after the same detailed routing for data No.1, respectively. The chip area is $137.2 \times 10^4\ \lambda^2$ and the total wire length is $62.0 \times 10^4\ \lambda$ by the proposed method, while by the conventional method, the chip area is $143.1 \times 10^4\ \lambda^2$ and the total wire length is $75.6 \times 10^4\ \lambda$ (see Data No.8 in Table 5.1). These results lead to the conclusion that FLORA II produces a better floorplan than the conventional one by comparing the final layout after the detailed routing.

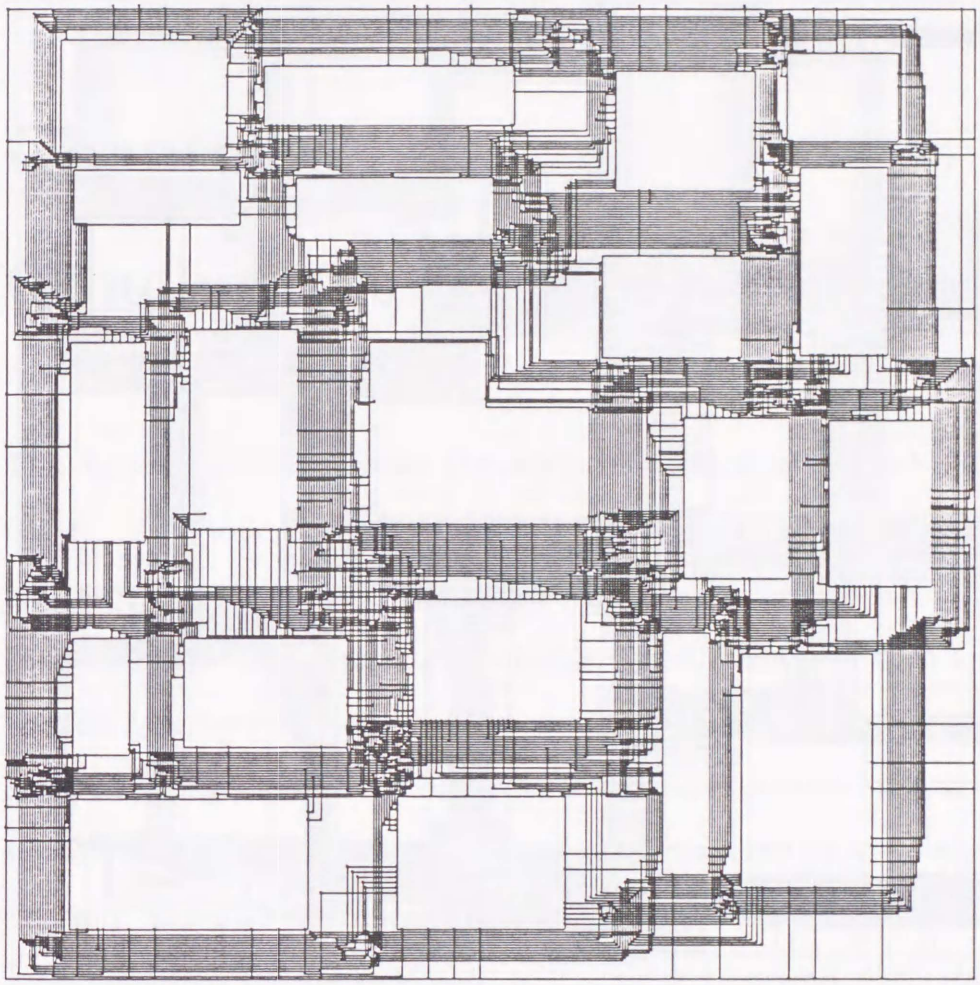


Fig. 5.3: Final layout by the proposed method.

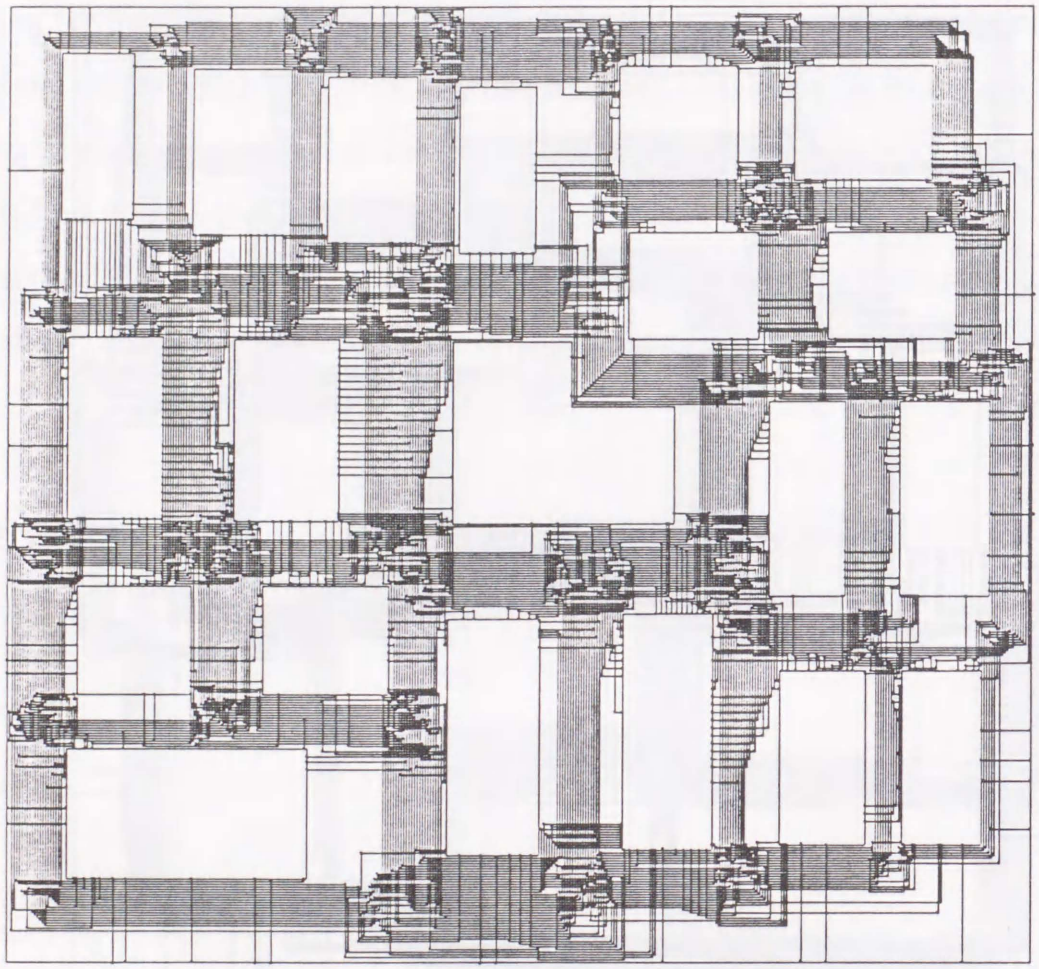


Fig. 5.4: Final layout by the conventional method.

Chapter 6

Conclusions

This dissertation has discussed hierarchical floorplanning for the VLSI layout design in building block approach. First, existing conventional floorplanning methods are briefly explained, and their drawbacks are pointed out. Secondly, a floorplanning problem is formally defined. Then a hierarchical floorplanning method, which merges floorplanning, global routing, and positioning together in a hierarchical fashion, is proposed to overcome these drawbacks. Thirdly, some related theoretical results are discussed. Fourthly, heuristic algorithms which are based on the theoretical results are proposed for the hierarchical floorplanning method with global routing and positioning. Each algorithm which plays an important role in the proposed floorplanning method is evaluated independently by the simulation experiments and verified its effectiveness. Finally, a prototype floorplanning system FLORA II based on the floorplanning method with global routing and positioning is developed on an AV310 workstation, and it is evaluated

by the simulation experiments on several data.

The main contributions of the study discussed in this dissertation are summarized as follows.

(1) Hierarchical floorplanning method combined with global routing and positioning

A hierarchical floorplanning method, which merges floorplanning and global routing together in a hierarchical fashion, was firstly proposed by W.-M. Dai et al. [Dai 86]. For this method, some drawbacks are pointed out, and a new floorplanning method which simultaneously determines a floorplan, *detailed global routes* which directly correspond to switchboxes and channels, and positioning is proposed in this dissertation, to overcome the drawbacks. By applying the proposed floorplanning method, a VLSI chip with small area and short wire length can be obtained.

(2) Theoretical results

Concerning the initial floorplanning process in the proposed hierarchical floorplanning the following theoretical results are obtained.

Ideal distance: For conventional initial placement methods of modules [Ueda 85] [Yamada 85], in which modules are regarded as points, several drawbacks are pointed out, and a concept called the *ideal distance* is introduced, in which the attractive effect and the repulsive effect are uniformly taken into account. Based on this concept, a heuristic algorithm is proposed and its effectiveness is verified by the simulation experiments.

Overlap resolution: During the overlap resolution phase, the optimality of the placement obtained in the previous phase should be preserved. This problem is formulated as an overlap resolution problem which preserves relative positions among modules, and proved to be *NP-hard.* Then a heuristic algorithm is proposed, and its effectiveness is verified by the simulation experiments.

One dimensional module placement: In VLSI layout design for building blocks, to place I/O modules optimally in the peripheral area of the chip is important to obtain a small chip. Placement problem for I/O modules is formulated as an improvement problem of one dimensional module placement, and an optimum algorithm of $O(|\mathrm{N}| \cdot (|\mathrm{M}| + log|\mathrm{N}|))$ computation time is proposed, where $|\mathrm{N}|$ is the number of nets and $|\mathrm{M}|$ is the number of modules. The correctness of the algorithm is proved, and some extension of the problem is also discussed.

(3) Proposal of the algorithms

In order to realize a proposed hierarchical floorplanning method, the following algorithms are proposed.

The routing-based partitioning: In our method, new channels which correspond to the division lines of modules are created so that the congestion of channels and switchboxes in the next level of hierarchy is reduced. This method is evaluated by the simulation experiments, and its effectiveness is verified.

The hierarchical detailed global routing: By giving hierarchy to the nets, and searching each global route on a global routing graph in a restricted region which is constructed around the corresponding route in the previous level of hierarchy, one of the optimal routes is succeeded from the previous level of hierarchy, and the correspondence between the global routes and the floorplan holds. The reduction of the computation time is also expected. This method is evaluated by the simulation experiments, and its effectiveness is verified.

The hierarchical positioning: In the proposed hierarchical positioning, the shapes of soft modules are adjusted to reduce dead spaces at each level of hierarchy. This method is applicable only in our floorplanning method, because floorplanning, global routing, and positioning are merged in a hierarchical fashion. This method is evaluated by the simulation experiments, and its effectiveness is verified.

(4) Application of the hierarchical floorplanning method with global routing and positioning

A prototype hierarchical floorplanning system FLORA II based on the proposed method is developed on an AV310 workstation, and is applied to several large data. From the results of the simulation experiments, the conclusion that the proposed hierarchical floorplanning method is sufficiently useful from a practical point of view is obtained.

Finally, future research works for the VLSI hierarchical floorplanning in

building block approach are discussed briefly.

(1) Development of the terminal positioner

After the proposed hierarchical floorplanning with global routing and positioning, the terminal positions of each net on the boundaries of soft modules must be determined instead of hypothetical terminals. In the reference [Ohmura 90a], a terminal positioning algorithm is proposed, in which the total wire length is minimized. By this method, however, each channel width is not optimized, and the area of the final layout may have dead space. Then, it is necessary to develop a new terminal positioning algorithm that minimizes the width of each channel.

(2) Improvement of the total layout system

The hierarchical floorplanning system FLORA II is extended to the total layout system by adding a conventional channel router and two conventional switchbox routers. Generally, it is difficult to complete the routing in a switchbox, and unexpected increase of chip area may occur during detailed routing. Replacing these channel router and the switchbox routers by more effective L-shaped channel [Dai 85] router, the problem mentioned above is reduced. A compaction which allows jog insertion is also effective to reduce dead space. In FLORA II, the following new positioning algorithm has the possibility to reduce the area of the floorplan and the final layout. In the hierarchical positioning, the topology of the global routing graph is preserved in order not to change the global routes at the same

level of hierarchy. By allowing the change of the topology of the global routing graph, and rerouting the global routes in the phase of hierarchical positioning, dead spaces may be reduced. In FLORA II, predesigned hard modules are allowed to have convex rectilinear polygons. But for virtual modules in each level of hierarchy, only the rectangular shapes are allowed. Introduction of the L-shaped soft module may also be effective.

Appendix: NP-hardness of Problem MPP4

[Problem MPP4] A logic circuit LC=(M,Te,N) and the placement region $[x_l, x_r]$ for modules are given as inputs.

Each net $n_i \in$ N consists of two terminals, one of which is a terminal t_j of a module and the another is an external terminal te_j. Then find the placement L(M) on x axis which satisfies the following condition i and condition ii, and minimizes the objective function $Z=\sum_{n_i \in N} c_i \cdot Zx(n_i)$, where c_i is the weight of the net n_i and $Zx(n_i)$ is the distance of two terminals on x axis.

(Condition i) Modules are not overlapped each other.

(Condition ii) Modules are placed in the placement region.

NP-hardness of this Problem MPP4 is proved as follows. In this proof, scheduling problem SS5 [0], which is known to be NP-complete, is shown to be reducible to the decision problem DP corresponding to the subproblem of Problem MPP4.

[Problem SS5]

Inputs:

(1) a set of tasks $T=\{ t_i \mid 1 \leq i \leq |T| \}$

(2) a length $l(t_i)$ for each $t_i \in T$

(3) a weight $we(t_i)$ for each $t_i \in T$

(4) a deadline $dd(t_i)$ for each $t_i \in T$

(5) a positive integer K

Question:

Is there a one-processor schedule σ for T which satisfies the following condition R1 ?

(Condition R1)

$\sum_{t_i \in Ta}(\sigma(t_i) + l(t_i) - dd(t_i)) \cdot we(t_i) \leq$ K,

where Ta= { t_i | $t_i \in$ T, and, $\sigma(t_i) + l(t_i) > dd(t_i)$ }

As a subproblem of Problem MPP4, the problem in which $[x_l, x_r]$ is restricted to [0, Lim]. The desision problem DR for this problem is defined as follows.

[Problem DP]

Inputs:

(1) a set of modules M= { M_i | 1 $\leq$ i $\leq$ | M | }

(2) a width $w(M_i)$ for each $M_i \in$ M

(3) a set of terminals T(M_i) = { tm_{2i-1}, tm_{2i} } for a module M_i, and the relative x coordinates $d(tm_i)$ for each $tm_i \in$ T(M_i)

(4) a set of external terminals Te= { te_i | 1 $\leq$ i $\leq$ | Te | }, and the x coordinates L(te_i) for each $te_i \in$ Te

(5) a netlist N (n_i = { tm_i, te_i }, where $n_i \in$N)

(6) a weight $c(n_i)$ for each $n_i \in$ N

(7) the placement region [0, Lim]

(8) a positive integer C

Question:

Is there a placement L(M) on x axis which satisfies the following conditions C1 - C3 ?

(Condition C1) Modules are not overlapped each other.

(Condition C2) Modules are placed in the placement region [0, Lim].

(Condition C3) $\sum_{M_i \in M} |\mathrm{L}(M_i) - \mathrm{L}(te_{2i-1})| \cdot c(n_{2i-1}) + |\mathrm{L}(M_i) + w(M_i) - \mathrm{L}(te_{2i})| \cdot c(n_{2i}) \leq \mathrm{C}$

[**Lemma A.1**] Decision problem DR is NP-complete.

(Proof) It is trivial that decision problem DR belongs to the class NP.

Polynomial transformation from Problem SS5 to Problem DR is shown. Assume that the inputs (1) - (5) are given as an instance of Problem SS5. Based on this, an instance of Problem DR is constructed as follows:

(1) a set of modules $\mathrm{M} = \{ M_i \mid t_i \in \mathrm{T} \}$

(2) a width $w(M_i) = l(t_i)$

(3) a set of terminals $\mathrm{T}(M_i) = \{ tm_{2i-1}, tm_{2i} \}$, where $d(tm_{2i-1})$ is 0 and $d(tm_{2i})$ is $l(t_i)$.

(4) a set of external terminals $\mathrm{Te} = \{ te_i \mid 1 \leq \mathrm{i} \leq \mid \mathrm{Te} \mid \}$, where $\mathrm{L}(te_{2i-1})$ is $-dd_i$), $\mathrm{L}(te_{2i})$ is dd_i),

(5)a netlist $\mathrm{N} = \{ n_i \mid 1 \leq \mathrm{i} \leq \mid \mathrm{N} \mid \}$, where $n_i = \{ tm_i, te_i$.

(6) a weight $c(n_{2i-1}) = c(n_{2i}) = we(t_i)$.

(7) $\mathrm{Lim} = (\mathrm{K} / \min \{ we(t_j) \}) + \max \{ dd(t_k) \}$

(8) a positive integer $\mathrm{C} = \sum_{t_i \in Ta} (2dd(t_i) - l(t_i)) \cdot we(t_i) + 2 \cdot \mathrm{K}$

Assume that the solution of Problem SS5 is true. A schedule σ which satisfies the condition R1 exists. Then $\mathrm{L}(M_i) = \sigma(t_i)$ $(t_i \in \mathrm{T})$.

First, the condition C1 is considered. It is obvious that the placement $\mathrm{L}(M_i) \geq 0$. From the condition R1, the following inequalities hold.

$(\sigma(t_i) + l(t_i) - dd(t_i)) \cdot we(t_i) \leq \mathrm{K}$,

$\sigma(t_i) + l(t_i) - dd(t_i) \leq K/min\{we(t_j)\}$ $(t_j \in T)$

$\sigma(t_i) + l(t_i) \leq (K/min\{we(t_j)\} + max\{dd(t_k)\}$ $(t_j \in T,\ t_k \in T)$

$\leq$ Lim

Then the condition C1 is satisfied.

Next, the condition C2 is considered. It is obvious that the condition C2 is satisfied.

Finally, the condition C3 is considered. From the condition R1, the following inequality holds.

$\sum_{t_i \in Ta}(\sigma(t_i) + l(t_i) - dd(t_i)) \cdot we(t_i) \leq \mathrm{K}$,

where Ta= $\{\ t_i \mid t_i \in \mathrm{T}$, and, $\sigma(t_i) + l(t_i) > dd(t_i)\ \}$ $\cdots (1)$

On the other hand, the left part of the inequality of the condition C3 can be transformed as follows.

$\sum_{M_i \in M} |\mathrm{L}(M_i) - \mathrm{L}(te_{2i-1})| \cdot c(n_{2i-1}) + |\mathrm{L}(M_i) + w(M_i) - \mathrm{L}(te_{2i})| \cdot c(n_{2i})$

$\leq \mathrm{C}$ $\cdots (2)$

Let $\mathrm{A}= |\mathrm{L}(M_i) - \mathrm{L}(te_{2i-1})| \cdot c(n_{2i-1}) + |\mathrm{L}(M_i) + w(M_i) - \mathrm{L}(te_{2i})| \cdot c(n_{2i})$,

$\mathrm{Q}= \{\ M_i \mid M_i \in \mathrm{M}$, and $\mathrm{L}(M_i) + w(M_i) > \mathrm{L}(te_{2i})\ \}$, and

$\bar{\mathrm{Q}} = \{\ M_i \mid M_i \in \mathrm{M}$, and $\mathrm{L}(M_i) + w(M_i) \leq \mathrm{L}(te_{2i})\ \}$.

Then the left part of the inequality (2) in represented as follows.

$$\text{(left part)} = \sum_{M_i \in \bar{Q}} \mathrm{A} + \sum_{M_i \in Q} \mathrm{A} \quad \cdots (3)$$

Because $c(n_{2i-1}) = c(n_{2i})$, the first term of the right part of the equation (3) is transformed as follows.

$$\sum_{M_i \in \bar{Q}} \mathrm{A} = \sum_{M_i \in \bar{Q}} (\mathrm{L}(M_i) - \mathrm{L}(te_{2i-1}) - \mathrm{L}(M_i) - w(M_i) + \mathrm{L}(te_{2i})) \cdot c(n_{2i})$$

$$= \sum_{M_i \in \bar{Q}} (\mathrm{L}(te_{2i}) - \mathrm{L}(te_{2i-1}) - w(M_i)) \cdot c(n_{2i})$$

The second term of the right part of the equation (3) is transformed as follows.

$$\sum_{M_i \in Q} \mathrm{A} = \sum_{M_i \in Q} (\mathrm{L}(M_i) - \mathrm{L}(te_{2i-1}) + \mathrm{L}(M_i) + w(M_i) - \mathrm{L}(te_{2i})) \cdot c(n_{2i})$$

$$= \sum_{M_i \in Q} (-\mathrm{L}(te_{2i}) - \mathrm{L}(te_{2i-1}) + w(M_i) + 2 \cdot \mathrm{L}(M_i)) \cdot c(n_{2i})$$

$$= \sum_{M_i \in Q} (\mathrm{L}(te_{2i}) - \mathrm{L}(te_{2i-1}) - w(M_i)) \cdot c(n_{2i})$$

$$+ \sum_{M_i \in Q} (-2 \cdot \mathrm{L}(te_{2i}) + 2 \cdot w(M_i) + 2 \cdot \mathrm{L}(M_i)) \cdot c(n_{2i})$$

$$= \sum_{M_i \in Q} (\mathrm{L}(te_{2i}) - \mathrm{L}(te_{2i-1}) - w(M_i)) \cdot c(n_{2i})$$

$$+ 2 \cdot \sum_{M_i \in Q} (\mathrm{L}(M_i) + w(M_i) - \mathrm{L}(te_{2i})) \cdot c(n_{2i})$$

Then

$$\text{(left part)} = \sum_{M_i \in \bar{Q}} \mathrm{A} + \sum_{M_i \in Q} \mathrm{A}$$

$$= \sum_{M_i \in \mathrm{M}} (\mathrm{L}(te_{2i}) - \mathrm{L}(te_{2i-1}) - w(M_i)) \cdot c(n_{2i})$$

$$+ 2 \cdot \sum_{M_i \in Q} (\mathrm{L}(M_i) + w(M_i) - \mathrm{L}(te_{2i})) \cdot c(n_{2i})$$

This equation can be transformed as follows.

$$\text{(left part)} = \sum_{t_i \in \mathrm{T}} (2 \cdot dd(t_i) - l(t_i)) \cdot we(t_i)$$

$$+ 2 \cdot \sum_{t_i \in \mathrm{Ta}} (\sigma(t_i) + l(t_i) - dd(t_i)) \cdot we(t_i)$$

From the equation (1) and the construction of the instance of Problem DR,

this equation is transformed to the following inequality.

(left part) $\leq \sum_{t_i \in Ta}(2dd(t_i) - l(t_i)) \cdot we(t_i) + 2 \cdot \mathrm{K}$

$\leq$ C (=(right part))

Then the condition C3 holds. Therefore, the solution of the instance of the problem DP is true.

Next, assume that the instance of Problem DR is true. Then the condition C3 holds.

(left part) $= \sum_{t_i \in \mathrm{T}} (2 \cdot dd(t_i) - l(t_i)) \cdot we(t_i)$

$+2 \cdot \sum_{t_i \in \mathrm{Ta}} (\sigma(t_i) + l(t_i) - dd(t_i)) \cdot we(t_i)$

(right part) $= \sum_{t_i \in Ta}(2dd(t_i) - l(t_i)) \cdot we(t_i) + 2 \cdot \mathrm{K}$

As (left part) $\leq$ (right part),

$\sum_{t_i \in Ta}(\sigma(t_i) + l(t_i) - dd(t_i)) \cdot we(t_i) \leq \mathrm{K}$,

where Ta= $\{ t_i \mid t_i \in \mathrm{T}$, and, $\sigma(t_i) + l(t_i) > dd(t_i) \}$

Then the condition R1 holds. Therefore, the solution of the instance of the problem SS5 is true.

[**Theorem A.1**] Problem MPP4 is $NP - hard$. □

References

[Asano 86] Asano, T.: *"A new compaction algorithm for building block packing technique,"* IECEJ Technical Report, CAS85-158 (1986).

[Chen 86] Chen, H. H. *and* Kuh, E. S.: *"Glitter: A gridless variable-width channel router,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-5, 4, pp.459-465 (1986).

[Chi 87] Chi., M. C.: *"An automatic rectilinear partitioning procedure for standard cells,"* Proc. 24th Design Automation Conf., pp.50-55 (1987).

[Ciesielski 87] Ciesielski, M. J. *and* Kinnen, E.: *"Digraph relaxation for 2-dimensional placement of IC blocks,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-6, 1, pp.55-66 (1987).

[Cohoon 88] Cohoon, J. P., Hegde, S. U., Martin, W. N., *and* Richards., D.: *"Floorplan design using distributed genetic algorithms,"* Proc. 1988 Int. Conf. on Computer-Aided Design (ICCAD-88), pp.452-455 (1988).

[Dai 85] Dai, W.-M., Asano, T., *and* Kuh, E. S.: *"Routing region definition and ordering scheme for building-block layout,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 4, 3, pp.189-197 (1989).

[Dai 86] Dai, W. M. *and* Kuh, E. S.: *"Hierarchical floor planning for buildin-block layout,"* Proc. 1986 Int. Conf. on Computer-Aided Design (ICCAD-86), pp.90-92 (1984).

[Dai 87a] Dai, W. M., Sato, M., *and* Kuh, E. S.: *"A dynamic and efficient representation of buildin-block layout,"* Proc. 24th Design Automation Conf., pp.376-384 (1987).

[Dai 87b] Dai, W.-M. *and* Kuh, E. S.: *"Simultaneous floorplanning and global routing for hierarchical building-block layout,"* IEEE Trans. on

Computer-Aided Design of Integrated Circuits and Systems, CAD-6, 5, pp.828-837 (1987).

[Dai 89] Dai, W.-M., Eschermann, B., Kuh, E. S., *and* Pedram, M.: *"Hierarchical placement and floorplanning in BEAR,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 8, 12, pp.1335-1349 (1989).

[Deutsch 76] Deutsch. D. N.: *"A dogleg channel router,"* Proc. 13th Design Automation Conf., pp.425-433 (1976).

[Eschermann 88] Eschermann, B., Dai, W. M., Kuh, E. S., *and* Pedram., M.: *"Hierarchical placement for macrocells: A "meet in the middle" approach,"* Proc. 1988 Int. Conf. on Computer-Aided Design (ICCAD-88), pp.460-463 (1988).

[Fiduccia 82] Fiduccia, C. M. *and* Matteyses, R. M.: *"A linear-time heuristic for improving network partitions,"* Proc. 19th Design Automation Conf., pp.175-181 (1982).

[Fujii 86] Fujii, T., Uchida, K., Ohmura, M., Kikuno, T., *and* Yoshida, N.: *"A new routing method based on tree-shaped region for VLSI layout design,"* Proc. 1986 Int. Symp. on Circuits and Systems (ISCAS 86), pp.319-320 (1986).

[Garey 79] Garey, M. R. *and* Johnson, D. S.: *"Computers and Intractability,"* W. H. Freedman and Company (1979).

[Hsu 87] Hsu, Y.-C. *and* Kubitz, W. J.: *"A procedure for chip floorplanning,"* Proc. 1987 Int. Symp. on Circuits and Systems (ISCAS 87), pp.568-571 (1987).

[Iwata 90] Iwata, A.: *"An overview of ASICs technologies,"* Journal of IEICE, 73, 10, pp.1032-1035 (1990) [in Japanese].

[Kleinhans 88] Kleinhans, J. M., Sigl, G., *and* Johannes., F. M.: *"GORDIAN: A new global optimization / rectangle dissection method for cell placement,"* Proc. 1988 Int. Conf. on Computer-Aided Design (ICCAD-88), pp.506-509 (1988).

[Kou 78] Koh, L., Markowsky, G., *and* Berman, L.: *"A fast algorithm for steiner trees,"* IBM Technical Report, RC-7390 (1978).

[Lai 88] Lai, Y.-T. *and* Leinwand, S. M.: *"Algorithms for floorplan design via rectangular dualization,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 7, 12, pp.1278-1289 (1988).

[LaPotin 86] LaPotin, D. P. *and* Director, S.W.: *"Mason: A global floorplanning approach for VLSI design,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-5, 4, pp.477-489 (1986).

[Lauther 79] Lauther, U.: *"A min-cut placement algorithm for general cell assembles based on a graph representation,"* Proc. 16th Design Automation Conf., pp.1-10 (1987).

[Leinwand 82] Leinwand, S. M. *and* Lai, Y. T.: *"An algorithm for building rectangular floor-plans,"* Proc. 21st Design Automation Conf., pp.663-664 (1984).

[Lin 89] Lin, Y.-L., Hsu, Y.-C., *and* Tsai, F.-S.: *"SILK: A simulated evolution router,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 8, 10, pp.1108-1114 (1989).

[Losleben 82] Losleben, P.: *"Computer Aided Design for VLSI,"* Chapter 4 in Very Large Scale Integration VLSI (Ed. Barbe, D. F.), Springer-Verlag (1982).

[Maling 82] Maling, K., Mueller, S. H., *and* Heller, W. R.: *"On finding most optimal rectangular pavkage plans,"* Proc. 19th Design Automation Conf., pp.663-670 (1982).

[Ohtsuki 86] Ohtsuki, T., ed.: *"Layout Design and Verification,"* Advanced in CAD for VLSI, vol. 4, North-holland, Amsterdam (1986).

[Ohmura 88] Ohmura, M., Izumoto, T., Fujii, T., Kikuno, T., *and* Yoshida, N.: *"A new floorplanning method with global routing based on functional partitioning,"* Proc. 1988 Int. Symp. on Circuits and Systems (ISCAS 88), pp.1697-1700 (1988).

[Ohmura 89] Ohmura, M., Miyao, J., Kikuno, T., *and* Yoshida, N.: *"Overlap resolution problem for block placement in VLSI layout,"* Trans. IEICE, J72-A, 7, pp.1093-1100 (1989) [in Japanese].

[Ohmura 90a] Ohmura, M., Toyohara, Y., Wakabayashi, S., Miyao, J., *and* Yoshida, N.: *"Determination of pin positions in VLSI floorplanning,"* Proceedings of 39th Annual Convention, IPSJ, 1V-7, pp.1626-1627 (1989) [in Japanese].

[Ohmura 90b] Ohmura, M., Wakabayashi, S., Miyao, J., *and* Yoshida, N.: *"Improvement of one dimensional module placement in VLSI layout design,"* Trans. IEICE, J73-A, 11, pp.1858-1866 (1990) [in Japanese].

[Ohmura 90c] Ohmura, M., Wakabayashi, S., Toyohara, Y., Miyao, J., *and* Yoshida, N.: *"Hierarchical floorplanning and detailed global routing with routing-based partitioning,"* Proc. 1990 Int. Symp. on Circuits and Systems (ISCAS 90), pp.1640-1643 (1988).

[Ohmura 91a] Ohmura, M., Wakabayashi, S., Miyao, J., *and* Yoshida, N.: *"Initial placement of modules based on ideal distance in VLSI layout design,"* Trans. IEICE, J74-A, 3, pp.576-579 (1991) [in Japanese].

[Ohmura 91b] Ohmura, M., Wakabayashi, S., Miyao, J., *and* Yoshida, N.: *"Hierarchical detailed floorplanning with global routing in VLSI layout design,"* Trans. IEICE, Section A, to appear.

[Otten 83] Otten, R. H. J. M.: *"Efficient floorplan optimization,"* Proc. Int. Conf. on Computer design (ICCD83), pp.499-502 (1983).

[Rivest 82] Rivest, R. L. *and* Fiduccia, C. M.: *"A "greedy" channel router,"* Proc. 19th Design Automation Conf., pp.418-424 (1982).

[Stockmeyer 82] Stockmeyer, L. J.: *"Optimal orientations of cells in slicing floorplan designs,"* IBM Technical Report, RJ-3731 (1982).

[Suaris 89] Suaris, P. R. *and* Kedem, G.: *"A quqadrisection-based combined place and route scheme for standard cells,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-5, 4, pp.477-489 (1986).

[Tsay 86] Tsay, R.-S. *and* Kuh, E. S.: *"A unified approach to circuit partitioning and placement,"* UC Berkeley Technical Report, UCB3731 (1982).

[Ueda 85] Ueda, K., Kitazawa, H., *and* Harada, I.: *"CHAMP: Chip floor plan for hierarchical VLSI layout design,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-4, 1, pp.12-22 (1985).

[Wang 90] Wang, T. C., *and* Wong, D. F.: *"An optimal algorithm for floorplan area optimization,"* Proc. 27th Design Automation Conf., pp.180-186 (1990).

[Watanabe 85] Watanabe, W., Asano, K., Kani, K., *and* Ohtsuki, T.: *"Design of VLSI I,"* Iwanami, Tokyo (1985).

[Watanabe 87] Watanabe, H. *and* Ackland, B.: *"FLUTE an expert floorplanner for full-custom VLSI design,"* IEEE Design & Test, 4, 1, pp.32-41 (1987).

[Wimer 88] Wimer, S., Koren, I., *and* Cederbaum, I.: *"Floorplans, planar graphs, and layouts,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 35, 3, pp.267-278 (1988).

[Wimer 89] Wimer, S., Koren, I., *and* Cederbaum, I.: *"Optimal aspect ratios of building blocks in VLSI,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 8, 2, pp.139-145 (1989).

[Wong 86] Wong, D. F. *and* Liu, C. L.: *"A new algorithm for floorplan design,"* Proc. 23rd Design Automation Conf., pp.101-107 (1989).

[Wong 89] Wong, D. F. *and* Sakhamuri, P. S.: *"Efficient floorplan area optimization,"* Proc. 26th Design Automation Conf., pp.586-589 (1989).

[Ying 89] Ying, C.-S. *and* Wong, J. S.-L.: *"An analytical approach to floorplanning for hierarchical building block layout,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 8, 4, pp.403-412 (1989).

[Yoshimura 82] Yoshimura, T. *and* Kuh, E. S.: *"Efficient algorithms for channel routing,"* IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-1, 1, pp.25-35 (1982).