WIRE ROUTING SCHEME

BASED ON

GRAPH THEORY MODEL

FEBRUARY 1977

TETSUO ASANO

WIRE ROUTING SCHEME
BASED ON
GRAPH THEORY MODEL

by

TETSUO ASANO

Submitted in partial fulfillment of

the requirement for the degree of

DOCTOR OF ENGINEERING

(Electrical engineering)

at

OSAKA UNIVERSITY

TOYONAKA, OSAKA, JAPAN

February 1977

ACKNOWLEDGEMENTS

The author would like to express his sincerest gratitude to Professor K. Tanaka, thesis supervisor. Prof. Tanaka is a constant source of invaluable advice and encouragement throughout the course of the research for this thesis. Special thanks are also due to Prof. T. Fujisawa, Prof. T. Kasami, and Prof. M. Kizawa of Department of Information and Computer Sciences through the course of author's undergraduate and graduate studies.

He wishes to thank Associate Prof. J. Toyoda, Associate Prof. S. Tamura, Dr. T. Kitahashi, Dr. M. Mizumoto, and the colleagues of Prof. Tanaka's laboratory for their several relevant suggestions and useful discussions.

He is also grateful to Mr. H. Horino and Mr. T. Amano of Central Research Laboratory, Hitachi Ltd., for their invaluable suggestions.

ABSTRACT

Tetsuo Asano, Graduate School of Osaka University, February 1977.

Wire Routing Scheme Based on Graph Theory Model: Doctoral Thesis.

The research described in this thesis deals with design automation of a building-block LSI. The principal aim has been to establish a wire-routing scheme superior to human designers. The algorithm has several features which sharply define it from previous techniques. Perfect wirability is the most noteworthy one among them. This enables it to design a layout pattern without any human supports.

This method realizes orthogonal wiring patterns on two layers; one for vertical routes called "trunks" and the other for horizontal routes called "branches". It first assigns each net to the simplest routing pattern with the minimum number of through-holes. The pattern consists of exactly one trunk and some branches. If these assignments fail to achieve an optimal pattern then it is necessary to break a net into more than one trunk. The author calls this operation a division of a trunk. A trunk is divided for one purpose of guaranteeing perfect wireability and for another purpose of reducing a wiring area.

The scheme with this divided-trunk style accomplishes a near optimal layout pattern, and has proven successful by experimental results.

LIST OF SYMBOLS

$\#A$:	number of elements of a set A
t_i:	trunk for ith net
$U(t_i)$:	y-coordinate of the upper end of the trunk t_i
$L(t_i)$:	y-coordinate of the lower end of the trunk t_i
$T_L(t_i)$:	set of y-coordinates of left-side terminals associated with the trunk t_i
$T_R(t_i)$	set of y-coordinates of right-side terminals associated with the trunk t_i
$\mathcal{R}$:	whole set of given nets or trunks
$R(t_i, t_j)$:	$T_L(t_i) \cap T_R(t_j)$, horizontal relation between two trunks
Γt_i:	$\{ t_j \mid R(t_i, t_j) = \emptyset \}$, set of those trunks which must be placed to the right of t_i
$\hat{\Gamma} t_i$:	set of trunks (vertices in HC-graph) reachable from t_i
$G_h(\mathcal{R}, \Gamma)$:	horizontal constraint graph, or HC-graph
$t_L(y_D)$:	trunk which connects with the left-side terminal at $y = y_D$
$t_R(y_D)$:	trunk which connects with the right-side terminal at $y = y_D$
$W(\mathcal{R})$:	minimum width necessary for arranging the trunks of $\mathcal{R}$, or track count of $\mathcal{R}$
$W_0(\mathcal{R})$:	maximum trunk-crossing count
$\hat{x}_L(t_i)$:	length of the longest constraint chain that terminates at t_i in HC-graph
$\hat{x}_R(t_i)$:	length of the longest constraint chain that starts from t_i in HC-graph

G_i: ith trunk group

$x_L(t_i)$: number of tracks which must be placed to the left of t_i

$x_R(t_i)$: number of tracks which must be placed to the right of t_i

(t_i, y_D, M): dividing pattern to divide a trunk t_i at $y = y_D$ by a method M

$R(t_i, y_D, M)$: set of trunks after dividing a trunk t_i at $y = y_D$ by a method M

$\mathcal{D}(R)$: decomposition of R

$G_v(R, A_v)$: vertical relation graph, or VR-graph

A_v: $\{ (t_i, t_j) \mid L(t_i) < U(t_j) \}$, set of directed arcs of VR-graph

$d_x(R)$: size of the optimum x-decomposition of R (x = a, c, w, r)

$d_x(R; S)$: size of the optimum x-decomposition of a subset S of R

$\tilde{x}_L(t_i)$: number of tracks necessary for arranging those trunks which must be placed to the left of t_i

$\tilde{x}_R(t_i)$: number of tracks necessary for arranging those trunks which must be placed to the right of t_i

$P(t_i, t_j)$: set of those trunks (vertices in HC-graph) which are on a directed path from t_i to t_j

$P_a(t_i, t_j)$: set of those trunks (vertices in HC-graph) through which every directed path from t_i to t_j must pass

$G_S(V_1, V_2, A_S)$: simplified VR-graph

$A \cup B$: union of mutually disjoint sets A and B

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Computer has enjoyed a steady growth since its first appearance in the mid-1940's. The remarkable development into the modern high-speed computer, however, had to wait until the introduction of integrated circuits (IC) or large-scale integration (LSI).

The recent trend in LSI technologies toward higher component density has drastically increased the complexity of designing a chip layout. From this point of view, development of a computer-aided design (CAD) system is urgently needed.

A design system for LSI may be divided into several parts --- partitioning a given network [1] - [4], placement of individual elements [5] - [7] and wire routing [8] - [18]. Among them, the most important is the last wire routing technique, since the time necessary for determining wiring routes takes more than half of the total design time.

It was in 1957 that R. C. Prim [8] and H. Loberman [9] suggested the methods of connecting terminals for the first time. In 1959, E. F. Moore [10] reported an algorithm for finding the shortest path through a maze. In the above works, the goal was laid on minimization of total wire length. In 1961, the most noticeable algorithm was presented by C. Y. Lee [11] (this algorithm is usually referred to as "Lee's algorithm"). It can find whatever sophisticated routes if any. Unfortunately, some serious disadvantages are also contained in this algorithm, i.e., too much storage required and too much time consumed.

For the last decade several modified versions of Lee's algorithm have been introduced: the algorithm for multi-layer boards of S. Heiss [12],

the line search technique of D. W. Hightower [13] or K. Mikami [14], the cellular routing method of Hitchcock [15], the stepping aperture technique of S. E. Lass [16], the channel routing technique of A. Hashimoto [17] and the method of S. B. Akers, Jr. [18].

In a layout design of LSI mask patterns, the first objective is the highest possible component density. The only way to accomplish this is to reduce an area for routing wires, since that for circuit cells is fixed. A building-block LSI is suitable for this purpose. It consists of several "blocks"; each block is composed of two parallel rows of circuit cells with an interconnection area between them. Such an LSI admits an approach in which a wiring pattern for an individual block is independently optimized. Then, what is an "optimal" wiring pattern? At the present state of the art, the optimality requires the following three: (1) Perfect wirability --- all of given nets are routed. (2) Minimum possible width --- nets in a block are routed within an area of minimum possible width. (3) Minimum possible number of through-holes --- a net is routed in the simplest style if possible.

Perfect wirability is indispensable for a design automation system, because its lack of the ability implies inevitability of human supports. Unfortunately, none of the wire routing techniques introduced above possesses it. Also, they may continue to make vain efforts to search for wiring routes even in the case where insufficient wiring area will not allow realization of required nets.

Optimal patterns may be found for a small-sized problem. But on the other hand in a practical, large-sized problem, the optimality is rather less practical. The author believes that a "good" wire routing program should be one of high cost/performance. This thesis directs the search

for a near optimal scheme for wire routing. The author takes an approach of iterative improvement; first assign each net to the simplest routing pattern, second guarantee perfect wireability with minimum possible modifications of routing patterns and last reduce the wiring area as much as possible. The improvement processes are based upon a channel routing style which is substantially more versatile than those previously analyzed. Specificially, previous papers [17] and [19] dealt with a style in which each net had a single, straight trunk with feeders to cell terminals; this thesis expands the style with methods of dividing a trunk, at any grid position, permitting the resulting subtrunks to overlap in the direction parallel to the terminal rows. The advantages of this expanded style are substantial: 1) trunks may be divided to eliminate cyclic horizontal constraints and 2) trunks may be divided to break up long constraint chains which cause the track count W to be larger than the maximum trunk-crossing count W_0 (sometimes $W \simeq 2 \cdot W_0$ for the single trunk style).

Chapter 2 presents a mathematical model and proves that the divided-trunk style can eliminate all of cyclic horizontal constraints except in some conditions which are expected to be extremely rare.

Chapter 3 is concerned with comparison between three dividing methods proposed and the conclusion is reached that the methods I and I' should be applied in preference to the other methods if possible. Then, extension of cycle elimination capability of these methods is intended. Also, an algorithm for eliminating all of cyclic horizontal constraints is presented.

Chapter 4 discusses a width reduction process. The algorithm proposed there takes a heuristic search method to achieve near optimal width. The main feature of the algorithm is that it can take advantage of a more

versatile routing style than those previously analyzed. In practice, this is an exhaustive process, so that it employs some heuristic functions to limit the search space.

CHAPTER 2

REALIZABILITY OF A SET OF NETS

2.1 Introduction

This chapter constructs a mathematical model for a wiring area, not far from a physical LSI mask pattern. On the model it is discussed whether a given set of nets is realizable.

A net is represented initially by the simplest routing pattern that consists of one trunk and some branches connecting it to appropriate terminals. Then, wire routing problem is equivalent to determining a layout pattern of trunks without overlapping branches of different nets. In order to avoid overlapping branches at any y-coordinate within the wiring area, a trunk with a branch contacting the left-side terminal must be located to the left of the other trunk with a branch contacting the right-side terminal. This thesis expresses such constraints between trunks by a directed graph. The author calls it a horizontal constraint graph, in short, an HC-graph.

A set of trunks should be realized by laying them out in the order determined by the corresponding HC-graph. Then, if the HC-graph contains any cycle, all of them must be eliminated. To accomplish this, the author devises only three simple methods of dividing a trunk and defines them formally. These formal definitions make it possible to prove that the divided-trunk style can eliminate cyclic constraints except in some conditions which are expected to be extremely rare.

2.2 Description of LSI Model

Fig. 2.1 illustrates a model for a mask pattern of a building-block

LSI. In order to describe the model used in this thesis, the following terms are presented.

(1) Element row: Row of fundamental circuit elements such as gates or flip-flops.

(2) Internal terminal row: Set of terminals in each element row. Each such set is assumed to form a straight line.

(3) Internal wiring area: Area between a pair of element rows.

(4) Block: Pair of element rows and an internal wiring area between them.

(5) External wiring area: Area for interconnection of different blocks.

(6) Upper and lower terminal rows: Rows of imaginary terminals at the upper and lower bounds, respectively, of a wiring area.

(7) Track: Straight line on which vertical wire segments run.

(8) Channel: General term for internal and external wiring areas.

As is seen in Fig. 2.1, the maskpattern of a building-block LSI consists of several blocks, external wiring areas and bonding pads. Wire routings are performed in internal and external wiring areas. Terminals of elements and through-holes are placed only at the quantized mesh points (grid points) on the maskpattern. Also, two layer wiring is permitted: one layer for vertical routes called trunks and the other for horizontal routes called branches. Trunks and branches are connected by through-holes.

In a building-block LSI, wire routings are performed first in internal wiring areas and then in external areas. This thesis assumes a model for a wiring area with terminal rows arranged in the y-direction, since such a model is valid for both an internal wiring area and external one. It

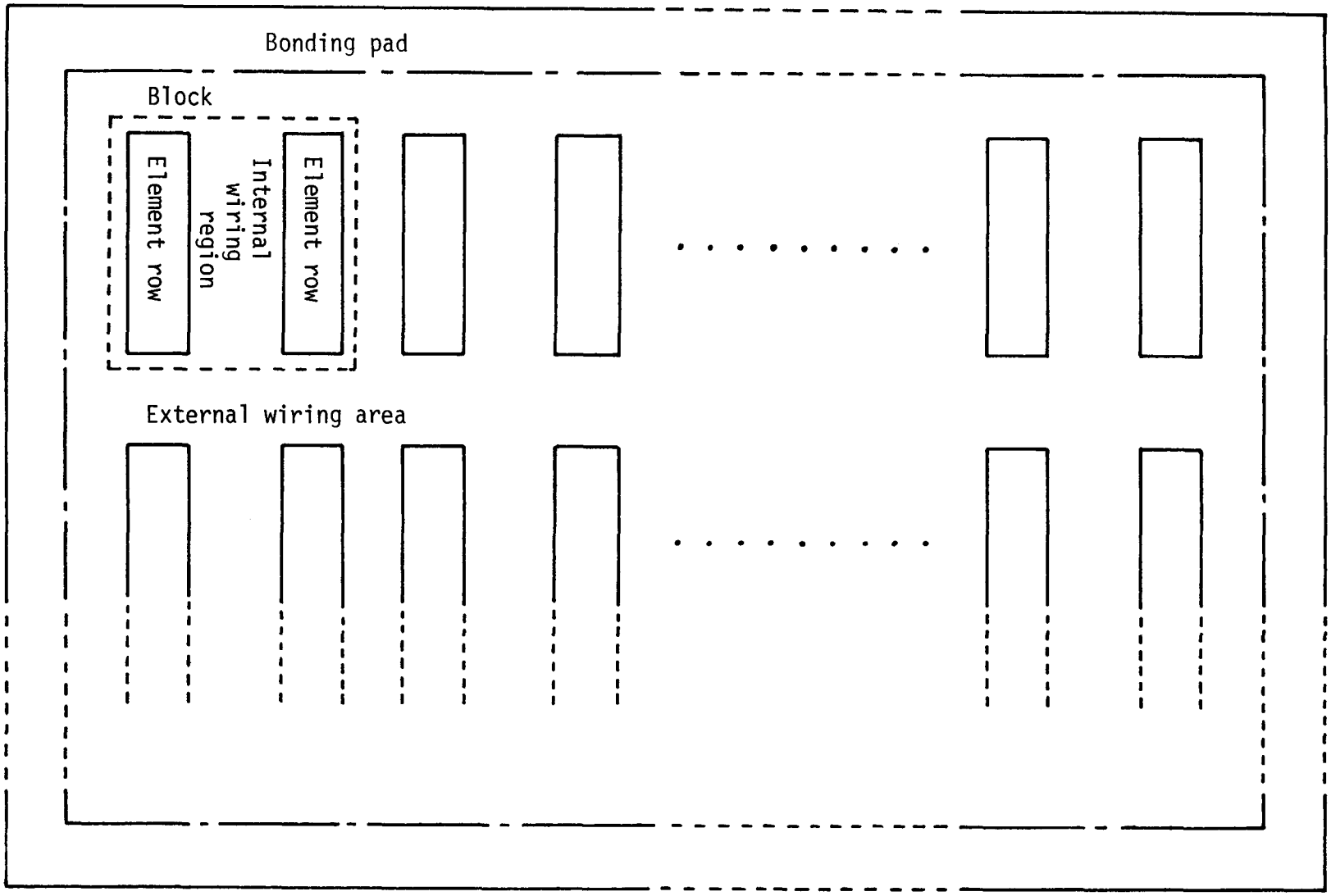


Fig. 2-1a. Whole structure of building-block LSI.

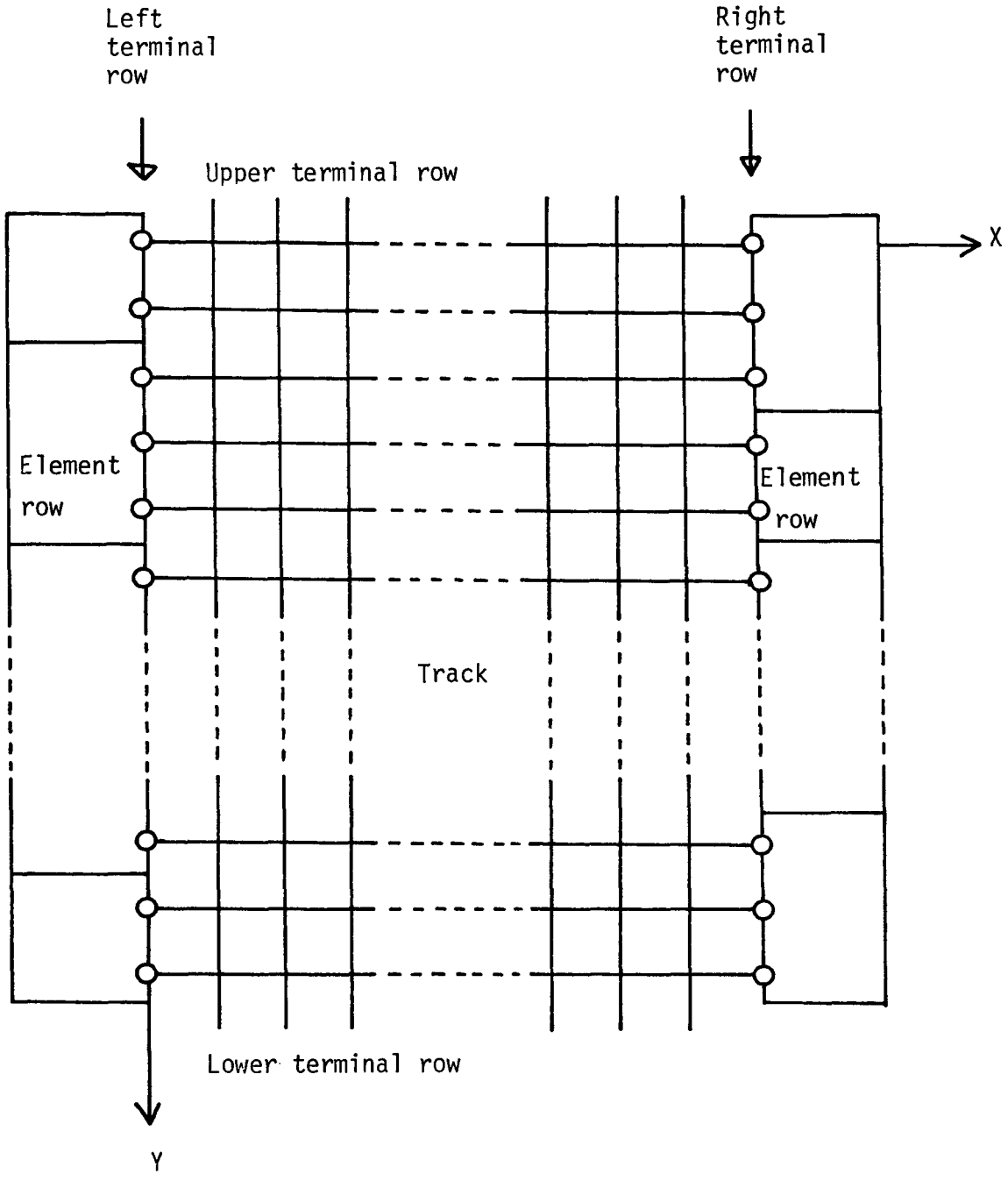


Fig. 2-1b. Block model.

is also assumed that the length of the left-side terminal row is equal to that of the right-side terminal row and that wire routings should be performed within the area.

Width W of a wiring area is defined by

W = (x-coordinate of the right terminal row) - 1,

where the coordinate is represented by the pitch number of meshes and the x-coordinate of the left terminal row is assumed to be zero.

2.3 Preliminary Definitions

In a layout design of a building-block LSI, the following design criteria are considered:

(1) Perfect wirability --- all of given nets are routed.

(2) Minimum possible width --- nets in a block are routed within an area of minimum possible width.

(3) Fewest possible through-holes --- a net is routed in the simplest possible style.

At the state of the art, total wire length is less significant.

A routed pattern is said to be "optimal" if it satisfies all the above criteria. Optimal patterns may be found for a small-sized problem. But on the other hand in a practical, large-sized problem, it seems impossible to find an optimal pattern. Moreover, even if one can construct a wire routing algorithm that always achieves an optimal pattern, it may not be valid to say that the algorithm is optimal. A practical wire routing algorithm should be evaluated by its cost/performance.

The wire routing scheme proposed in this thesis has a structure of iterative improvement. In this sense the process should start from the

simplest routing style. Fig. 2.2 shows the simplest routing pattern of a net i on a quantized wiring area. This wiring route consists of one vertical line segment called a trunk, and some horizontal line segments called branches which connect it to appropriate terminals. Here it should be noted that exactly one trunk is assigned to the net. The net may be identified with the trunk associated with the net. This implies that input data are specifications of trunks. Hereafter, let t_i denote the trunk for the ith net.

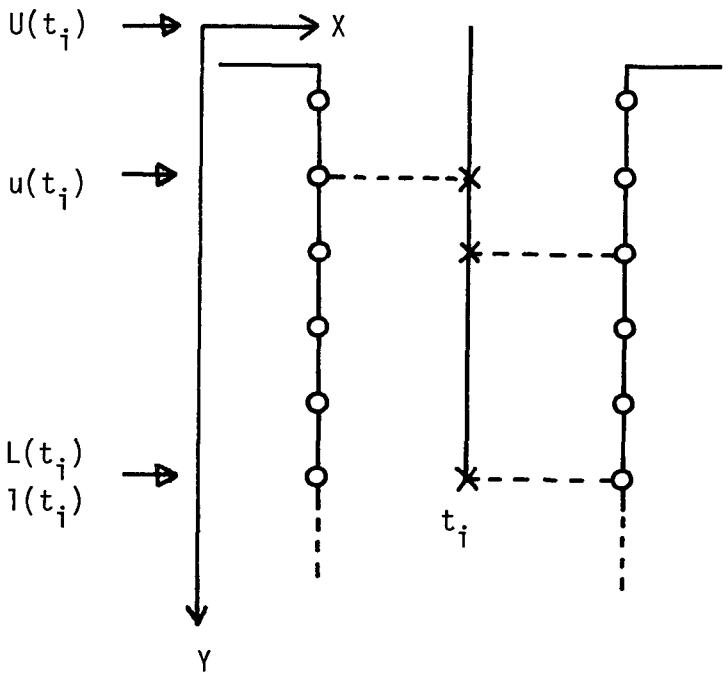


Fig. 2-2. Simplest wiring pattern for net i.

The first formalization step in this wiring scheme is a set representation of required nets. The trunk t_i is specified by two values $U(t_i)$ and $L(t_i)$, and two sets $T_L(t_i)$ and $T_R(t_i)$, where $T_L(t_i)$ is the set of y-

coordinates of left-side terminals related to the ith net, and $T_R(t_i)$ for right-side terminals. $U(t_i)$ and $L(t_i)$ are the upper end and the lower end of the trunk t_i, respectively. Formally, they are designated as follows:

$$U(t_i) = \begin{cases} 0 & \text{if the trunk } t_i \text{ goes through upwards,} \\ u(t_i) & \text{otherwise.} \end{cases}$$

$$L(t_i) = \begin{cases} \infty & \text{if the trunk } t_i \text{ goes through downwards,} \\ \ell(t_i) & \text{otherwise.,} \end{cases}$$

where $u(t_i) = \min(T_L(t_i) \cup T_R(t_i))$ and $\ell(t_i) = \max(T_L(t_i) \cup T_R(t_i))$.

The second formalization step is a directed-graph representation of constraints between trunks. A wire routing program specifies positions of trunks in order. These specifications must be done so that they do not overlap branches of different nets. In order to avoid overlapping branches at any y-coordinate, a trunk t_i with a branch contacting the left-side terminal must be located to the left of the other trunk t_j with a branch contacting the right-side terminal. This thesis expresses such a horizontal constraint between these two trunks t_i and t_j as $t_j \in \Gamma t_i$ (see Fig. 2.3). Formally, the set Γt_i contains t_j if and only if the intersection $T_L(t_i) \cap T_R(t_j)$ is not empty. Γ may be considered as a multiple-valued function or mapping from the set of trunks R into itself.

It is convenient to consider the inverse mapping Γ^{-1} of Γ as given by

$$\Gamma^{-1} t_i = \{ t_j \mid t_i \in \Gamma t_j \}.$$

For a set of trunks A, ΓA and $\Gamma^{-1} A$ are defined by

$$\Gamma A = \bigcup_{t_i \in A} \Gamma t_i, \text{ and}$$

$$\Gamma^{-1} A = \bigcup_{t_i \in A} \Gamma^{-1} t_i.$$

Transitive closures $\hat{\Gamma}$ and $\hat{\Gamma}^{-1}$ of Γ and Γ^{-1}, respectively, are defined by

$$\hat{\Gamma}t_i = \Gamma t_i \cup \Gamma^2 t_i \cup \cdots, \text{ and}$$

$$\hat{\Gamma}^{-1}t_i = \Gamma^{-1}t_i \cup \Gamma^{-2}t_i \cup \cdots,$$

where for each k, $k \geq 1$

$$\Gamma^{k+1}t_i = \Gamma(\Gamma^k t_i), \text{ and}$$

$$\Gamma^{-k-1}t_i = \Gamma^{-1}(\Gamma^{-k}t_i).$$

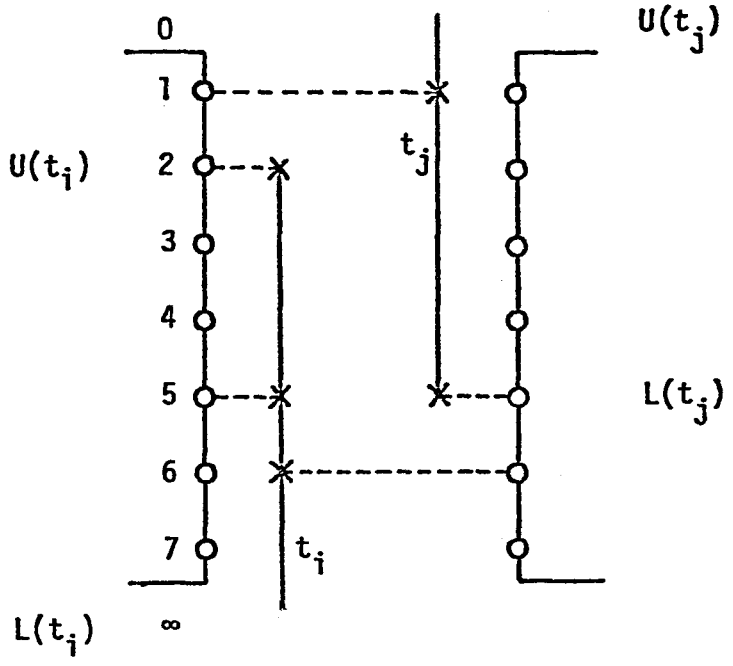


Fig. 2. Horizontal constraint between two trunks.

$T_L(t_i) = \{2, 5\}$, $T_R(t_i) = \{6\}$, $U(t_i) = 2$, $L(t_i) = \infty$,

$T_L(t_j) = \{1\}$, $T_R(t_j) = \{5\}$, $U(t_j) = 0$, $L(t_j) = 5$,

$R(t_i, t_j) = \{5\}$, $t_j \in \Gamma t_i$.

A horizontal constraint graph, in short, an HC-graph, $G_h = (R, \Gamma)$ is formed as follows: Each vertex of G_h corresponds to a trunk and a directed arc is drawn from a vertex t_i to a vertex t_j if and only if $t_j \in \Gamma t_i$ holds. An HC-graph does not contain a self-loop (an arc with the same vertex at its start and end point) or a parallel arc.

Throughout this thesis a cycle in an HC-graph is represented as a set of vertices (trunks). This representation should cause no confusion, since an HC-graph has no parallel arcs in it.

Example 2.1 A set of nets is given in a tabular form as Table 2.1.

As is seen from the table, the scheme introduced in this thesis identifies a set of nets with the corresponding set of trunks. Various kinds of optimization which may be required in the course of the process can be achieved by altering the way of assigning trunks to nets.

As for the set of trunks $R_{2,1}$, horizontal relations between trunks are

$$\Gamma t_1 = \{t_2\},\ \Gamma t_2 = \{t_4\},\ \Gamma t_3 = \{t_1, t_5\},\ \Gamma t_4 = \{t_5\} \text{ and } \Gamma t_5 = \emptyset.$$

The above relations are mapped into the HC-graph shown in Fig. 2.4. Fig. 2.4 indicates that the set of trunks can be realized by placing the trunk t_3 on the leftmost track, t_1 on the next track and so on.

On the other hand, the HC-graph for the set of trunks given in Table 2.2 contains a cycle $\{t_1, t_2\}$, as is seen in Fig. 2.5. This implies that the trunk t_1 must be laid to the left of t_2 and also t_2 must be laid to the left of t_1, which is impossible.

The above suggests the following theorem, which is the most fundamental theorem on realizability of a set of trunks.

Theorem 2.1 The necessary and sufficient condition that a set of trunks R can be arranged without any overlap of branches is that the HC-graph

corresponding to R does not contain any cycles.

(Proof omitted)

This theorem states that if any cycles are contained in the HC-graph then all of them must be eliminated by dividing trunks in suitable ways. This should be done by representing a net by two or more trunks. The author calls such an operation "a division of a trunk". Methods of dividing a trunk are proposed in the next section.

t_i	$T_L(t_i)$	$T_R(t_i)$	$U(t_i)$	$L(t_i)$
t_1	{1}	{2}	0	4
t_2	{4}	{1}	1	4
t_3	{2, 5}	$\emptyset$	2	∞
t_4	{3}	{4}	0	3
t_5	$\emptyset$	{3, 5}	3	∞

Table 2.1. A set of nets $R_{2,1}$.

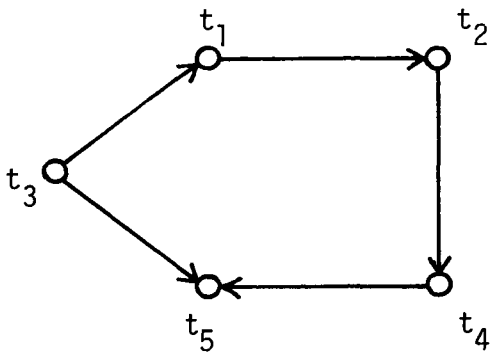


Fig. 2-4. HC-graph for the set of trunks $R_{2,1}$.

t_i	$T_L(t_i)$	$T_R(t_i)$	$U(t_i)$	$L(t_i)$
t_1	{1}	{4}	0	4
t_2	{4}	{1}	1	4
t_3	{2, 5}	$\emptyset$	2	∞
t_4	{3}	{2}	0	3
t_5	$\emptyset$	{3, 5}	3	∞

Table 2.2. A set of nets $R_{2,2}$.

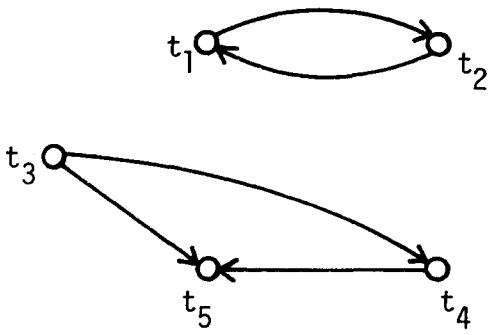


Fig. 2-5. HC-graph for the set of trunks $R_{2,2}$.

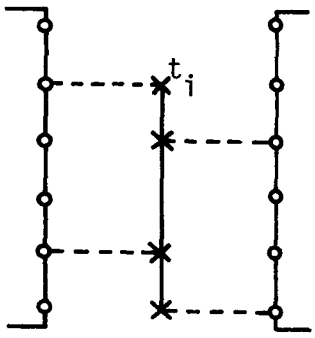
2.4 Trunk Division Methods

This section describes trunk-division methods. This scheme limits them to only three kinds shown in Fig. 2.6. These three kinds of methods are enough for achieving perfect wirability, which is shown in Section 2.6.

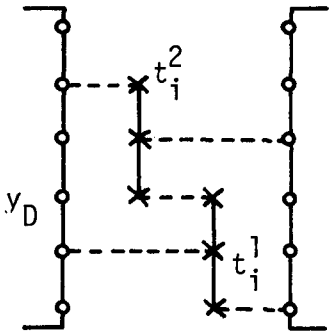
These methods perform trunk-division operations as follows: Method I divides a trunk into two trunks, one above the dividing point and the other below it. The upper trunk is arranged to the left of the lower trunk. Method I' is the dual form of Method I.

The other methods produce two trunks, one for connections to the left-side terminals and the other for connections to the right-side terminals. These two trunks are connected by a branch at a dividing point. The point of Method II is above or below all of the terminals associated with the trunk to be divided. Method III divides a trunk at one of its own right-side terminals, and Method III' at one of its own left-side terminals.

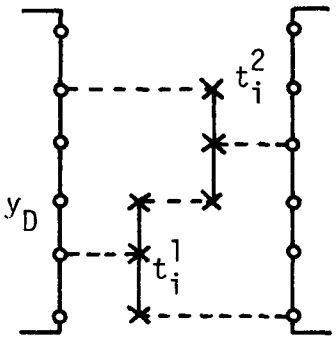
The three trunk-division methods are formally expressed as follows. In Fig. 2.6-(2), for instance, the divided trunk t_i^2 connects with two branches. Notice that one connects to the right-side terminal while the other, i.e., the branch with the y-coordinate being y_D, does not. In the expressions below the existence of such a branch at a dividing point (y_D) is reflected by the overscored y-coordinate, such as $\bar{y}_D$. For the trunk t_i^2, $\bar{y}_D$ is added to the right set $T_R(t_i^2)$. Such a representation leads to the following formal expressions of the three methods. In the expressions below, a trunk t_i is divided at $y = y_D$ by each of the methods into two distinct trunks t_i^1 and t_i^2, and the left and right sets of t_i^1 and t_i^2 are specified.



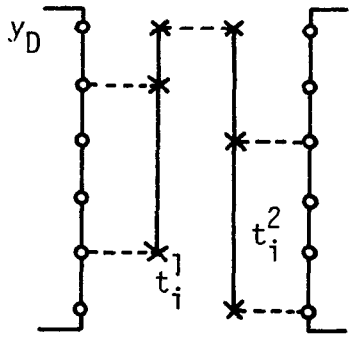
(1) Original pattern



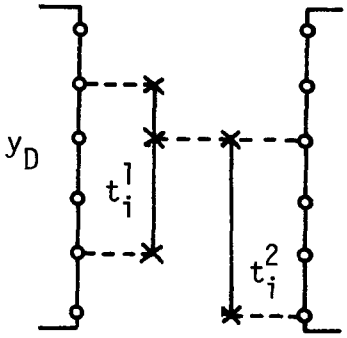
(2) Method I



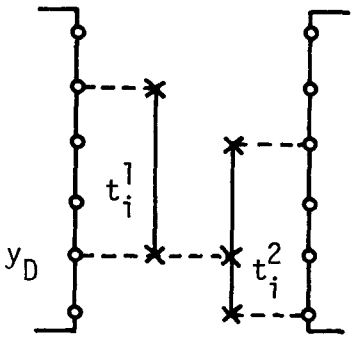
(3) Method I'



(4) Method II



(5) Method III



(6) Method III'

Fig. 2-6. Three methods of dividing a trunk.

Method I

$$T_L(t_i^1) = \{ y \mid y \in T_L(t_i) \text{ and } y > y_D \} \cup \{ \bar{y}_D \},$$

$$T_R(t_i^1) = \{ y \mid y \in T_R(t_i) \text{ and } y \geqq y_D \},$$

$$T_L(t_i^2) = \{ y \mid y \in T_L(t_i) \text{ and } y \leqq y_D \},$$

$$T_R(t_i^2) = \{ y \mid y \in T_R(t_i) \text{ and } y < y_D \} \cup \{ \bar{y}_D \}.$$

The condition for its application is that $U(t_i) < y_D < L(t_i)$.

Method I'

$$T_L(t_i^1) = \{ y \mid y \in T_L(t_i) \text{ and } y \geqq y_D \},$$

$$T_R(t_i^1) = \{ y \mid y \in T_R(t_i) \text{ and } y > y_D \} \cup \{ \bar{y}_D \},$$

$$T_L(t_i^2) = \{ y \mid y \in T_L(t_i) \text{ and } y < y_D \} \cup \{ \bar{y}_D \},$$

$$T_R(t_i^2) = \{ y \mid y \in T_R(t_i) \text{ and } y \leqq y_D \}.$$

The condition for its application is the same as that of Method I.

Method II

$$T_L(t_i^1) = T_L(t_i), \qquad T_R(t_i^1) = \{ \bar{y}_D \},$$

$$T_L(t_i^2) = \{ \bar{y}_D \}, \qquad T_R(t_i^2) = T_R(t_i).$$

The condition for its application is

(1) $T_L(t_i) \neq \emptyset$ and $T_R(t_i) \neq \emptyset$, and

(2) $L \geqq y_D > \ell(t_i)$ or $U \leqq y_D < u(t_i)$,

where L is the lower end and U is the upper end of the wiring area.

Method III

$$T_L(t_i^1) = T_L(t_i), \qquad T_R(t_i^1) = \{ \bar{y}_D \},$$

$$T_L(t_i^2) = \{ \bar{y}_D \}, \qquad T_R(t_i^2) = T_R(t_i) \quad .$$

The condition for its application is

(1) $T_L(t_i) \neq \emptyset$, (2) $y_D \in T_R(t_i)$ and (3) $\#T_R(t_i) \geqq 2$.

Method III'

$$T_L(t_i^1) = T_L(t_i), \qquad T_R(t_i^1) = \{ \bar{y}_D \},$$

$$T_L(t_i^2) = \{ \bar{y}_D \}, \qquad T_R(t_i^2) = T_R(t_i).$$

The condition for its application is

(1) $T_R(t_i) \neq \emptyset$, (2) $y_D \in T_L(t_i)$ and (3) $\#T_L(t_i) \geq 2$.

Here, for a set A, #A denotes the number of elements of A.

The following is the definition of Γ modified so that Γ should be able to represent newly generated arcs.

Definition 2.1 For a set $T_X(t_i) = \{y_1, y_2, \ldots, y_m, \bar{y}_D\}$ (X = L or R), let $T_X^0(t_i) = \{y_1, y_2, \ldots, y_m\}$ and let $T_X^1(t_i) = \{ y_D \}$. Then, for two trunks t_i and t_j the relation $R(t_i, t_j)$ is defined by

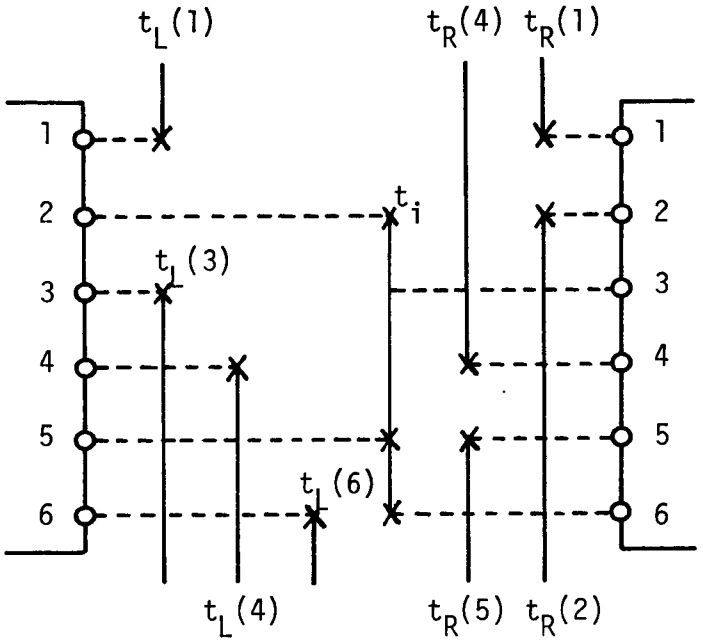
$$R(t_i, t_j) = \begin{cases} \emptyset & \text{if } t_i = t_j, \\ (T_L^0(t_i) \cap T_R^0(t_j)) \cup (T_L^0(t_i) \cap T_R^1(t_j)) & \\ \quad \cup (T_L^1(t_i) \cap T_R^0(t_j)) \cup (T_R^1(t_i) \cap T_L^1(t_j)) & \text{otherwise,} \end{cases}$$

and Γ is defined by

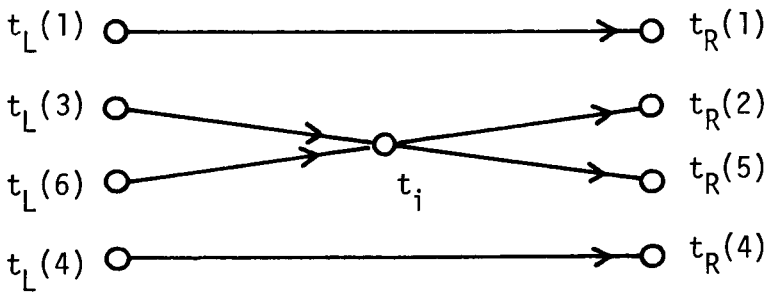
$$\Gamma t_i = \{ t_j \mid R(t_i, t_j) \neq \emptyset \}.$$

In order to illustrate the effects of dividing a trunk upon an HC-graph, a situation shown in Fig. 2.7 is assumed, where the set of trunks $\mathcal{R}$ is $\{t_L(1), t_L(3), t_L(4), t_L(6), t_i, t_R(1), t_R(2), t_R(4), t_R(5)\}$.

In the situation the trunk t_i is divided by each of the methods. Then, the HC-graph is altered as shown in Fig. 2.8, where $\mathcal{R}(t_i, y_D, M)$ is the set of trunks after the trunk t_i is divided at $y = y_D$ by the method M.

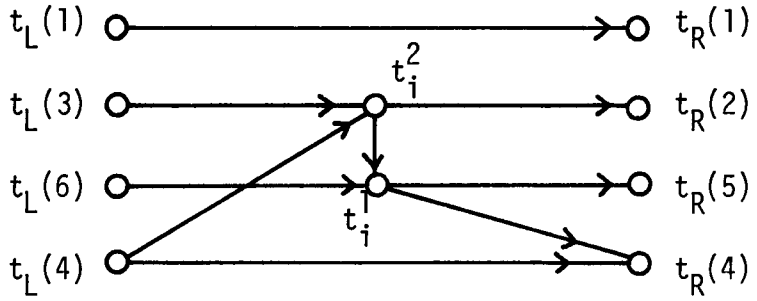


(a) Wiring pattern

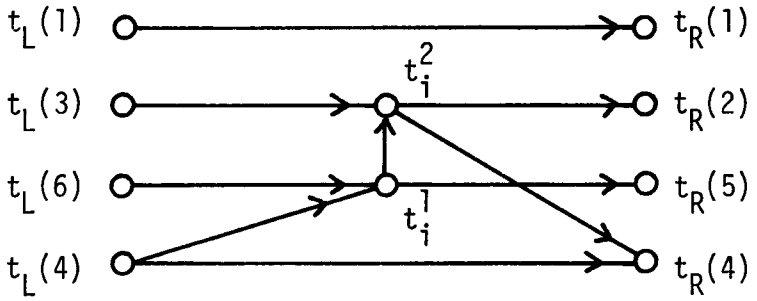


(b) HC-graph

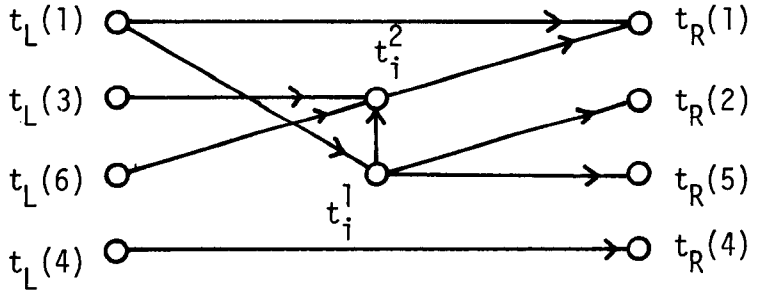
Fig. 2-7. Surroundings of the trunk t_i.



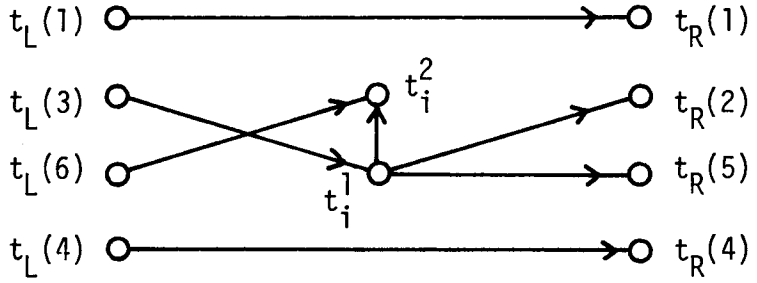
(1) HC-graph for R(t_i, 4, I)



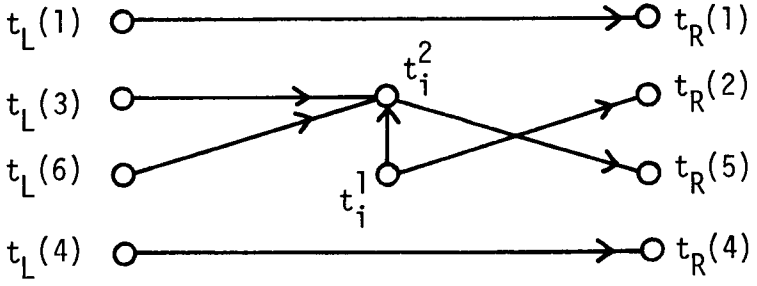
(2) HC-graph for R(t_i, 4, I')



(3) HC-graph for R(t_i, 1, II)



(4) HC-graph for R(t_i, 3, III)



(5) HC-graph for R(t_i, 5, III')

Fig. 2-8. The effect of each trunk-division method.

2.5 Elimination of Cyclic Horizontal Constraints

This section examines conditions under which a cycle in an HC-graph can be eliminated by dividing a trunk by each kind of the methods proposed in the previous section.

2.5.1 Methods I and I'

Definition 2.2 Let $\mathcal{R}$ be a set of trunks. Then, for every point y_j, $t_X(y_j)$ (X = L or R) is defined as

$$t_X(y_j) = \begin{cases} t_k & \text{if there exists } t_k \in \mathcal{R} \text{ such that } T_X(t_k) \ni y_j, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where "undefined" means that the terminal at $y = y_j$ on the corresponding side is not associated with any nets. Such a terminal is called an empty terminal.

In general, dividing a trunk t_i at $y = y_D$ by the method I generates the arcs $(t_i^1, t_R(y_D))$, $(t_L(y_D), t_i^2)$ and (t_i^2, t_i^1) if $t_L(y_D)$ and $t_R(y_D)$ are defined and not identical with the trunk t_i. If $t_L(y_D)$ and/or $t_R(y_D)$ are undefined or equal to t_i, the arcs including them are not generated. On the other hand, for the method I', generated arcs are $(t_i^2, t_R(y_D))$, $(t_L(y_D), t_i^1)$ and (t_i^1, t_i^2).

Definition 2.3 In an HC-graph G_h, three consecutive vertices (trunks) t_{i-1}, t_i, t_{i+1} such that $\Gamma t_{i-1} \ni t_i$ and $\Gamma t_i \ni t_{i+1}$ are said to be separable at $y = y_D$ if they satisfy

(SC-A) $\min R(t_{i-1}, t_i) > y_D > \max R(t_i, t_{i+1})$,

or

(SC-B) $\min R(t_i, t_{i+1}) > y_D > \max R(t_{i-1}, t_i)$.

Throughout this section the following assumptions are made without loss of generality.

(A1) A cycle C is of the form $\{t_1, t_2, \ldots, t_n = t_1\}$, where $n \geq 3$

and $\Gamma t_j \ni t_{j+1}$ for $j = 1, 2, \ldots, n-1$.

(A2) $T_L^1(t_i) = T_R^1(t_i) = \emptyset$ for each trunk $t_i \in C$, that is, C contains no divided trunks.

Lemma 2.1 A cycle C is eliminated by dividing a trunk $t_i \in C$ at $y = y_D$ by the method I or I' if and only if t_{i-1}, t_i, t_{i+1} are separable at $y = y_D$.

(Proof) Suppose that t_{i-1}, t_i and t_{i+1} satisfy the separability condition (SC-A) of Definition 2.3: Divide the trunk t_i at $y = y_D$ by the method I. Then, from (SC-A) and (A2), we have

$$R(t_{i-1}, t_i) = T_L(t_{i-1}) \cap T_R(t_i) \subseteqq T_R(t_i^1)$$

and

$$R(t_i, t_{i+1}) = T_L(t_i) \cap T_R(t_{i+1}) \subseteqq T_L(t_i^2).$$

Hence, we have

$$t_i^1 \in \Gamma t_{i-1},\ t_i^2 \notin \Gamma t_{i-1},\ t_i^1 \notin \Gamma^{-1} t_{i+1} \text{ and } t_i^2 \in \Gamma^{-1} t_{i+1}.$$

This indicates that the cycle C has been eliminated. Further, in this case it is easily shown that any division of t_i by the method I' does not succeed in eliminating the cycle C.

In case where t_{i-1}, t_i and t_{i+1} satisfy the separability condition (SC-B), dividing t_i at $y = y_D$ by the method I' leads to

$$t_i^1 \notin \Gamma t_{i-1},\ t_i^2 \in \Gamma t_{i-1},\ t_i^1 \in \Gamma^{-1} t_{i+1} \text{ and } t_i^2 \notin \Gamma^{-1} t_{i+1},$$

which mean that the cycle C has been opened.

The necessary part is straightforward from the above discussion.

Q. E. D.

Theorem 2.2 A cycle C is eliminated without generating any new cycles by dividing a trunk $t_i \in C$ at $y = y_D$ by the method I or I' if and only if

(1) t_{i-1}, t_i and t_{i+1} are separable at $y = y_D$

and

(2) one of the following holds:

(i) $y_D \in T_L(t_i) \cup T_R(t_i)$,

(ii) neither $t_L(y_D)$ nor $t_R(y_D)$ is defined,

(iii) $t_L(y_D)$ only is defined and $t_L(y_D) \notin \hat{\Gamma} t_i$ holds,

(iv) $t_R(y_D)$ only is defined and $t_R(y_D) \notin \hat{\Gamma}^{-1} t_i$ holds, or

(v) both $t_L(y_D)$ and $t_R(y_D)$ are defined and $t_L(y_D) \notin \hat{\Gamma} t_i$, $t_R(y_D) \notin \hat{\Gamma}^{-1} t_i$, $t_L(y_D) \notin \hat{\Gamma} t_R(y_D)$ and $t_L(y_D) \neq t_R(y_D)$ hold.

(Proof) The proof proceeds by contradiction. It must be shown that if t_{i-1}, t_i and t_{i+1} are separable or if none of the conditions (i) - (v) of the theorem is satisfied then the division either leaves the cycle C or produces some new cycles. It has been known in the previous lemma that if the separability condition is not the case then the cycle C can not be eliminated. Thus it is sufficient to show that if t_{i-1}, t_i and t_{i+1} are separable at $y = y_D$ but if none of the conditions from (i) to (v) holds then the division produces some new cycles. Here, one may assume (SC-A) of the separability condition, since the other condition (SC-B) is the dual form of (SC-A) and hence the proof is similar. In this case the method I must be selected as mentioned in the proof of Lemma 2.1.

The negation of the condition (iii) means that there exists at least one of the trunks $t_L(y_D)$ and $t_R(y_D)$. Also, it is known by negating (i) that those trunks differ from the trunk t_i to be divided. Thus, it is enough to consider the following cases:

(a) $t_L(y_D)$ is defined and $t_L(y_D) \in \hat{\Gamma} t_i$,

(b) $t_R(y_D)$ is defined and $t_R(y_D) \in \hat{\Gamma}^{-1} t_i$,

(c) both $t_L(y_D)$ and $t_R(y_D)$ are defined and $t_L(y_D) \in \hat{\Gamma} t_R(y_D)$, and

(d) both $t_L(y_D)$ and $t_R(y_D)$ are defined and $t_L(y_D) = t_R(y_D)$.

It is shown in the below that if one of the above conditions holds then the division produces a new cycle. First, the case (a) is considered. Since the method I was adopted, the arc $(t_L(y_D), t_i^2)$ has been generated. After the division of the trunk t_i, at least one of the relations $t_L(y_D) \in \hat{\Gamma} t_i^1$ and $t_L(y_D) \in \hat{\Gamma} t_i^2$ holds because $t_L(y_D) \in \hat{\Gamma} t_i$ in the original HC-graph. If $t_L(y_D) \in \hat{\Gamma} t_i^1$ holds then this division generates a cycle $t_i^1 \rightarrow \cdots \rightarrow t_L(y_D) \rightarrow t_i^2 \rightarrow t_i^1$ and if $t_L(y_D) \in \hat{\Gamma} t_i^2$ does then a cycle $t_i^2 \rightarrow \cdots \rightarrow t_L(y_D) \rightarrow t_i^2$, both of which are new cycles since the original HC-graph contains no arc corresponding to $(t_L(y_D), t_i^2)$ (see Fig. 2.9(a)).

The case (b) can be proved in a similar manner.

Next, in the case (c), the division generates a cycle $t_i^2 \rightarrow t_i^1 \rightarrow t_R(y_D) \rightarrow \cdots \rightarrow t_L(y_D) \rightarrow t_i^2$, which is new for the same reason as in the case (a) (see Fig. 2.9(b)).

Finally, the case (d) may be regarded as a special case of (c).

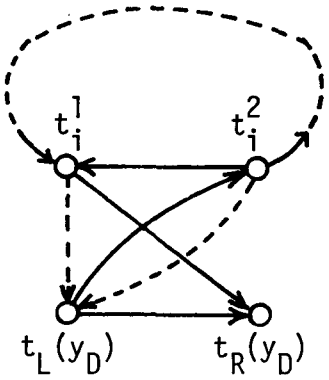
The sufficient part is obvious from the above.

Q. E. D.

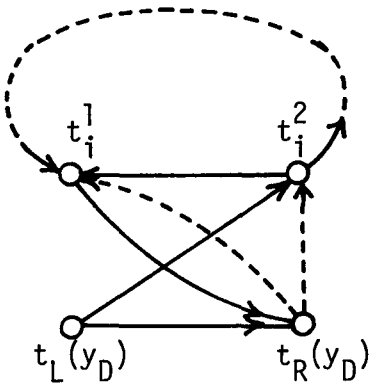
As an example, consider the set of trunks given in Table 2.2. The HC-graph shown in Fig. 2.5 contains a cycle $\{t_1, t_2\}$. Divide, for instance, the trunk t_1 at $y = 2$ by the method I, and the resulting HC-graph contains no cycle. In this case the trunk $t_L(2)$ is t_3 and the trunk $t_R(2)$ is t_4, both of which satisfy the condition (2) stated in Theorem 2.2.

2.5.2 Method II

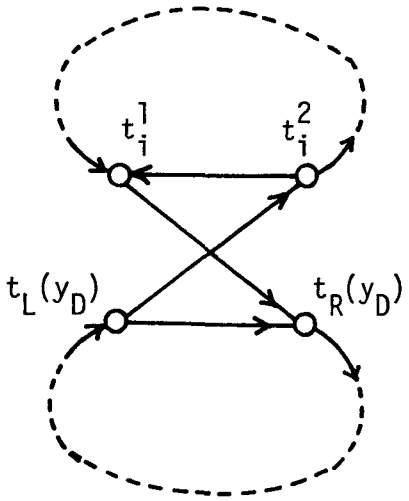
Theorem 2.3 A cycle C is eliminated without generating any new cycles by dividing a trunk $t_i \in C$ at $y = y_D$ by the method II if and only if



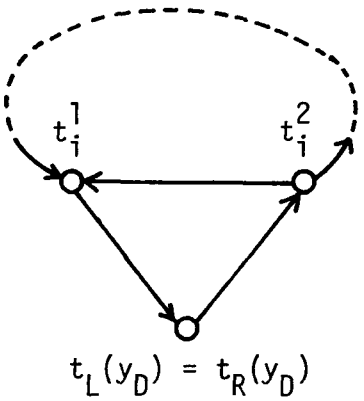
(a) Case (a)



(b) Case (b)



(c) Case (c)



(d) Case (d)

Fig. 2.9. Generation of a new cycle.

(1) $L \geqq y_D > \ell(t_i)$ or $U \leqq y_D < u(t_i)$

and

(2) one of the following holds:

(i) neither $t_L(y_D)$ nor $t_R(y_D)$ is defined,

(ii) $t_L(y_D)$ only is defined and $t_L(y_D) \notin \hat{\Gamma} t_i$,

(iii) $t_R(y_D)$ only is defined and $t_R(y_D) \notin \hat{\Gamma}^{-1} t_i$,

(iv) both $t_L(y_D)$ and $t_R(y_D)$ are defined and $t_L(y_D) \notin \hat{\Gamma}^{-1} t_i$, $t_R(y_D) \notin \hat{\Gamma}^{-1} t_i$, $t_L(y_D) \notin \hat{\Gamma} t_R(y_D)$ and $t_L(y_D) \neq t_R(y_D)$.

(Proof omitted)

2.5.3 Methods III and III'

The following theorem states that the methods III and III' proposed have the most powerful ability to eliminate cycles. So those trunks to which the method III or III' is applicable should be distinguished from others, and they are called "mighty trunks".

Theorem 2.4 A cycle C is eliminated without generating any new cycles by dividing some trunks in C by the method III or III' if and only if C contains any mighty trunks.

(Proof) The proof follows from the following two lemmas.

Lemma 2.2 If a cycle C contains two consecutive trunks t_i and t_{i+1} such that $\#T_L(t_i) \neq \#T_R(t_{i+1})$, it can be eliminated by dividing t_i or t_{i+1} by the method III or III'.

(Proof) Suppose $\#T_L(t_i) > \#T_R(t_{i+1})$. There is an element y_D such that $y_D \in T_L(t_i)$ and $y_D \notin T_R(t_{i+1})$. Dividing the trunk t_i at $y = y_D$ by the method III', we have

$$\Gamma^{-1} t_i^1 = \{ t_j \mid R(t_j, t_i^1) \neq \emptyset \}$$

$$= \{ t_j \mid T_L^0(t_j) \cap T_R^1(t_i^1) \neq \emptyset \text{ and } t_j \neq t_i^1 \} = \emptyset$$

and $t_{i+1} \notin \Gamma t_i^2$, since $R(t_i^2, t_{i+1}) = \emptyset$.

The above shows that the cycle C has been eliminated.

In case $\#T_L(t_i) < \#T_R(t_{i+1})$, dividing the trunk t_{i+1} at $y = y_D$ by the method III where $y_D \in T_R(t_{i+1}) - T_L(t_i)$ eliminates the cycle C.

Q. E. D.

(Remark) Division by either of the methods III and III' generates no new cycles since it is always true for such divisions that

$$\Gamma t_i^1 \cup \Gamma t_i^2 \subseteq \Gamma t_i \cup \{t_i^1, t_i^2\}$$

and

$$\Gamma^{-1} t_i^1 \cup \Gamma^{-1} t_i^2 \subseteq \Gamma^{-1} t_i \cup \{t_i^1, t_i^2\}.$$

Lemma 2.3 If a cycle C contains two consecutive trunks t_i and t_{i+1} such that $\#T_L(t_i) \geq 2$ and $\#T_R(t_{i+1}) \geq 2$, it can be eliminated by dividing t_i and/or t_{i+1} by the methods III and/or III'.

(Proof) If $T_L(t_i) \not\subseteq T_R(t_{i+1})$, there exists an element y_D such that $y_D \in T_L(t_i) - T_R(t_{i+1})$. Then, dividing the trunk t_i at $y = y_D$ by the method III' eliminates the cycle C as mentioned in the preceding lemma. Also, if $T_R(t_{i+1}) \not\subseteq T_L(t_i)$, dividing the trunk t_{i+1} by the method III leads to the desired results.

Next, suppose $T_L(t_i) = T_R(t_{i+1}) = S$; it follows from the assumption of this lemma that S contains at least two distinct elements y_1 and y_2. Divide the trunk t_i at $y = y_1$ by the method III' and t_{i+1} at $y = y_2$ by the method III. Then it should be noted that the trunks t_i^1 and t_{i+1}^2 should not be contained in any cycles in the resulting HC-graph since $\Gamma^{-1} t_i = \emptyset$ and $\Gamma t_{i+1}^2 = \emptyset$. Also, one can easily see that $\Gamma t_i^2 \not\ni t_{i+1}^1$ because $T_L^0(t_i^2) = T_R^0(t_{i+1}^1) = \emptyset$ and $T_R^1(t_i^2) = T_L^1(t_{i+1}^1) = \emptyset$. Thus, all the above shows that the cycle C has been eliminated.

Q. E. D.

2.6 Realizability of a Set of Nets

The preceding sections were concerned with the problem of how to eliminate a given cycle by applications of the proposed methods. This section considers whether a given set of nets can be realized, in other words, all of cycles in a given HC-graph can be eliminated, under the constraints listed below:

(1) Two-layer wiring is permitted. One layer is used for horizontal wiring routes and the other for vertical routes.

(2) Wire routing is always performed in a meshwise way and no half-mesh wiring is permitted.

(3) All of wiring routes must be within a wiring area whose vertical length is fixed.

The constraints above are commonly applicable to LSI fabrication. There is no restriction on the width W of the wiring area, which will be considered in the following chapters.

Theorems 2.2, 2.3 and 2.4 are used below to find the condition to be satisfied by a given HC-graph in order for all cycles to be eliminated.

Definition 2.4 Let G be a directed graph. (i) A pair of vertices v_i and v_j of G are connected if $\Gamma v_i \ni v_j$ or $\Gamma v_j \ni v_i$. (ii) Let V be a set of vertices of G which are connected and let v_k be a vertex of G such that $v_k \notin V$. Then, the union $V \cup \{ v_k \}$ is connected if there exists some vertex v_i in V such that $\Gamma v_i \ni v_k$ or $\Gamma v_k \ni v_i$.

Definition 2.5 A component of a graph is a subgraph whose set of vertices is maximally connected.

Definition 2.6 Let C_1 and C_2 be cycles. If there exists $t_i \in C_1$ and $t_j \in C_2$ such that $t_j \in \hat{\Gamma} t_i$ and $t_i \in \hat{\Gamma} t_j$, then C_1 and C_2 are said to be

connected, and if C_1 and C_2 do not have such a pair of trunks, they are said to be independent.

Components of an HC-graph are classified into the following three.

(1) Cycle-free component: Component containing no cycle.

(2) Loop component: Component containing cycles but no mighty trunks.

(3) Compound component: Component containing cycles and at least one mighty trunk.

Lemma 2.4 A loop component consists of only one cycle.

(Proof) By the definition of a loop component, it has at least one cycle in it. Let it be $C = \{t_1, t_2, \ldots, t_n = t_1\}$, where $n \geq 3$ and $\Gamma t_j \ni t_{j+1}$, $j = 1, 2, \ldots, n-1$. Since no mighty trunks are contained, $\#T_L(t_i) = \#T_R(t_i) = 1$ for any trunk $t_i \in C$. Thus, $\Gamma t_i = \{t_{i+1}\}$ and $\Gamma^{-1} t_i = \{t_{i-1}\}$ hold, so that this component is composed only of the cycle C.

Q. E. D.

Lemma 2.5 All of cycles in a compound component can be eliminated.

(Proof) First, note those cycles in a compound component which are mutually independent. Each of these cycles contains at least one mighty trunk, since otherwise the component will be a loop component. Hence, it is known from Theorem 2.4 that all of these cycles can be eliminated by applications of the methods III and III'.

The above states that it is enough to deal with mutually connected cycles in the component. Let $\hat{C}$ be union of cycles which are connected with one another and independent of others. Since $\hat{C}$ is composed of more than one cycle, each cycle contains mighty trunks. Consider a mighty trunk t_i to which the method III is applicable. Divide t_i by the method III at $y = y_D$ such that $y_D \in T_R(t_i)$, and we have

$$\#\Gamma^{-1}t_i^1 = \#\{\ t_p \mid R(t_p, t_i^1) \neq \emptyset\ \}$$

$$= \#\{\ t_p \mid T_L^0(t_p) \cap T_R^1(t_i^1) \neq \emptyset \text{ and } t_p \neq t_i^1\ \}$$

$$= \#\{\ t_p \mid y_D \in T_L^0(t_p) \text{ and } t_p \neq t_i^1\ \}$$

$$\leq 1,$$

and

$$\#\Gamma t_i^2 = \#\{\ t_q \mid R(t_i^2, t_q) \neq \emptyset\ \}$$

$$= \#\{\ t_q \mid T_L^1(t_i^2) \cap T_R^0(t_q) \neq \emptyset \text{ and } t_q \neq t_i^2\ \}$$

$$= \#\{\ t_q \mid y_D \in T_R^0(t_q) \text{ and } t_q \neq t_i^2\ \}$$

$$= 0.$$

The above means that if there exists a trunk t_i in $\hat{C}$ such that $\#\Gamma^{-1}t_i \geq 2$ then it can be replaced with the trunks t_i^1 and t_i^2 such that $\#\Gamma^{-1}t_i^1 \leq 1$ and $\#\Gamma t_i^2 = 0$. Here t_i^2 does not belong to any cycle. If this operation failed to eliminate all the cycles in $\hat{C}$, those cycles which remained to be eliminated would be mutually independent since $\#\Gamma^{-1}t_p \leq 1$ for any trunk t_p of such any cycle. If any cycles should be left in the resulting set of $\hat{C}$, all of them can be eliminated, for each of them still contains those trunks to which the method III' is applicable. An example is t_q such that $\#T_R(t_q) = 1$ and $\#T_L(t_q) \geq 2$ which remains undivided by the above operation and another is a trunk t_i^1 for some divided trunk t_i. So the lemma has been proved.

Q. E. D.

From the results obtained we can derive the most important theorem in this thesis.

Theorem 2.5 A set of nets is unrealizable if and only if it has all the following properties:

(1) No terminal is empty. In other words, every terminal is associated with some net.

(2) For any trunk t_i, $\#T_L(t_i) = \#T_R(t_i) = 1$.

(3) There exists a trunk t_i such that $T_L(t_i) \neq T_R(t_i)$.

(Proof) ("if" part) Consider the ith net: Property (1) and property (2) guarantee that $\#\Gamma t_i = \#(T_L(t_i) - T_R(t_i))$ and $\#\Gamma^{-1}t_i = \#(T_R(t_i) - T_L(t_i))$ in the HC-graph, since every terminal is associated with some net. Therefore, if $T_L(t_i) \neq T_R(t_i)$ then $\#\Gamma t_i = \#\Gamma^{-1}t_i = 1$ and otherwise $\#\Gamma t_i = \#\Gamma^{-1}t_i = 0$, i.e., the trunk t_i corresponds to an isolated vertex in the HC-graph. Property (3) guarantees that the HC-graph contains vertices that are not isolated. It follows that the HC-graph corresponding to a set of nets having the properties of this theorem is composed of several loop components, maybe together with some isolated vertices. In order to eliminate a cycle comprising a loop component, a trunk of the cycle must be divided by one of the methods I, I' and II, whereas it is evident that, wherever divided, the conditions of Theorem 2.2 or 2.3 should not be satisfied.

("only if" part) The proof proceeds by contradiction. First, consider the case where the left terminal at $y = y_0$ is empty and the right terminal at $y = y_0$ is not. Then the trunk t_0 containing y_0 in its right set does not belong to a loop component but some other. Since compound components which may be contained in the HC-graph can be reduced to a cycle-free component, as verified in Lemma 2.5, it is enough to deal with the case where the HC-graph contains loop components C_1, C_2, ... , C_n and a cycle-free component C_0 containing the trunk t_0. It is shown below that the cycles C_1, C_2, ... , C_n can be eliminated successively. Consider a trunk t_i of the component C_1 such that $\max T_L(t_i) <$

min $T_R(t_i)$. Since $y_0 \notin T_L(t_i) \cup T_R(t_i)$ from the assumption, there are three cases to consider:

(1) $\max T_L(t_i) < y_0 < \min T_R(t_i)$,

(2) $y_0 < \max T_L(t_i) < \min T_R(t_i)$, and

(3) $\max T_L(t_i) < \min T_R(t_i) < y_0$.

In the case of (1) the method I can be applied to the trunk t_i at $y = y_0$. Here, the conditions of Theorem 2.2 are satisfied since $t_L(y_0)$ is undefined and $t_R(y_0) = t_0 \notin \hat{\Gamma}^{-1}t_i$. Therefore, the division of t_i eliminates the cycle C_1. On the other hand, in the cases (2) and (3) the method II can be applied to the trunk t_i at $y = y_0$. Hence, the cycle C_1 is eliminated by the division of t_i. Here it should be noted that a new cycle-free component including C_0 and C_1 has resulted from this division. In other words, we have come back to the starting point of the proof. Therefore, C_2, C_3, ... , C_n can be successively eliminated.

Easier is the proof in case that both of the terminals at $y = y_0$ are empty.

Secondly, if each trunk t_i has the property that $T_L(t_i) = T_R(t_i)$, the set of trunks is evidently realizable.

Lastly, consider the case where the condition (3) of this theorem does not hold. Here it may be further assumed that there exist no empty terminals. There are three cases to consider. In either case it is sufficient to show, as mentioned above, that all of those loop components can be eliminated which may be contained in the HC-graph.

Case 1: Let t_i be a trunk such that $\#T_L(t_i) = 0$, i.e., $\Gamma t_i = \emptyset$ (its dual case can be proved in an entirely similar manner). Since $T_R(t_i) \neq \emptyset$ and there are no empty terminals, we have $\Gamma^{-1}t_i \neq \emptyset$. Thus,

the trunk t_i can not belong to a loop component, that is, there is a non-loop component containing at least two trunks. Further, it may be assumed that this component is cycle-free since Lemma 2.5 guarantees that any compound component can be reduced to a cycle-free component. Consequently, as mentioned earlier, even if the HC-graph contains any loop components, their cycles can be successively eliminated.

Case 2: Suppose that there is a trunk t_i such that $\#T_L(t_i) \geq 2$ and $T_L(t_i) \neq T_R(t_i)$. Then, the trunk t_i belongs to a non-loop component, which has at least two vertices since

$$\#\Gamma t_i = 0 \text{ if and only if } \#(T_L(t_i) - T_R(t_i)) = 0.$$

Therefore the remaining proof proceeds similarly to the case 1.

Case 3: Suppose that the HC-graph contains a trunk t_i such that $\#T_L(t_i) \geq 2$ and $T_L(t_i) = T_R(t_i)$. Divide the trunk t_i by the method III as follows:

$$T_L(t_i^1) = T_L(t_i), \qquad T_R(t_i^1) = \{ \bar{y}_D \},$$

$$T_L(t_i^2) = \{ \bar{y}_D \}, \qquad T_R(t_i^2) = T_R(t_i),$$

where $y_D \in T_L(t_i) = T_R(t_i)$. Then, we have

$$\Gamma t_i^1 = \{ t_i^2 \} \text{ and } \Gamma^{-1} t_i^1 = \emptyset,$$

$$\Gamma t_i^2 = \emptyset \qquad \text{and } \Gamma^{-1} t_i^2 = \{ t_i^1 \}.$$

This shows that the component $\{t_i^1, t_i^2\}$ is cycle-free, as required. This terminates the proof of this theorem.

Q. E. D.

In a practical problem, a set of nets satisfying all of the conditions of Theorem 2.5 is expected to be extremely rare. It follows that we can derive the conclusion that any set of nets is realizable by using the methods proposed in this thesis.

Example 2.2 Consider a set of nets given in Table 2.3.

t_i	$T_L(t_i)$	$T_R(t_i)$	$U(t_i)$	$L(t_i)$
t_1	{3, 9}	{1, 8}	1	9
t_2	{13}	{9}	9	13
t_3	{2, 15}	{3, 13}	2	15
t_4	{4, 16}	{6, 15}	4	16
t_5	{8}	{2, 16}	2	16
t_6	{1, 11}	{4}	1	11
t_7	{6}	{11}	6	11
t_8	{5}	{12}	5	12
t_9	{14}	{5}	5	14
t_{10}	{12}	{14}	12	14
t_{11}	{7}	{10}	7	10
t_{12}	{10}	{7}	7	10

Table 2.3. A set of nets $R_{2,3}$.

The HC-graph corresponding to the above set of nets is shown in Fig. 2.10, which is composed of three components: one is a compound component and the other two are loop components. As is easily seen, the HC-graph contains nine cycles $\{t_1, t_3, t_5\}$, $\{t_1, t_2, t_3, t_5\}$, $\{t_1, t_2, t_3, t_4, t_5\}$, $\{t_1, t_3, t_4, t_5\}$, $\{t_1, t_2, t_3, t_4, t_6\}$, $\{t_1, t_3, t_4, t_6\}$, $\{t_4, t_6, t_7\}$, $\{t_8, t_9, t_{10}\}$ and $\{t_{11}, t_{12}\}$.

First of all, three trunks t_3, t_4 and t_5 are divided in order to eliminate all cycles in the compound component: t_3 at $y = 2$ by the

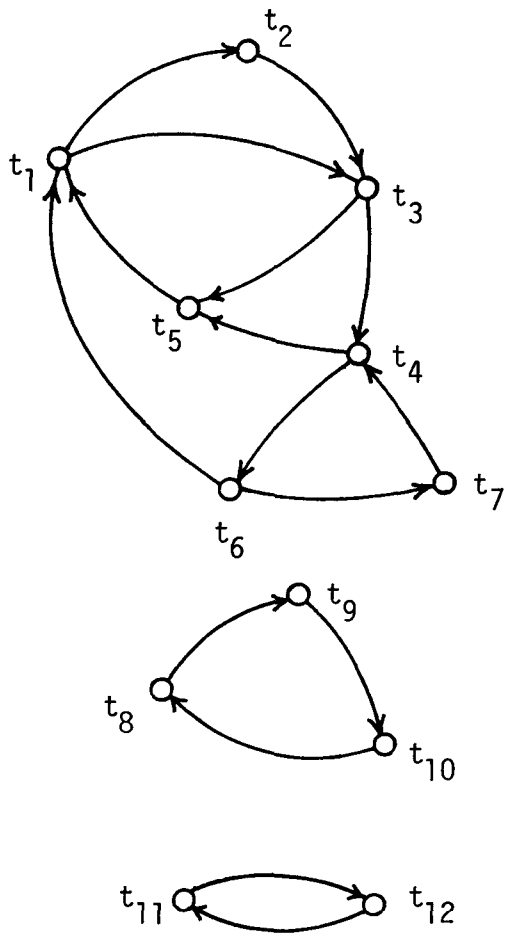


Fig. 2.10. Original HC-graph.

method III', t_4 at y = 15 and t_5 at y = 16 both by the method III. Then all the cycles of this component are eliminated as shown in Fig. 2.11. There still remain two cycles $\{t_8, t_9, t_{10}\}$ and $\{t_{11}, t_{12}\}$. Divide the trunk t_8 at y = 11 by the method I and the trunk t_{11} at y = 12 by the method II, and the HC-graph results as shown in Fig. 2.12.

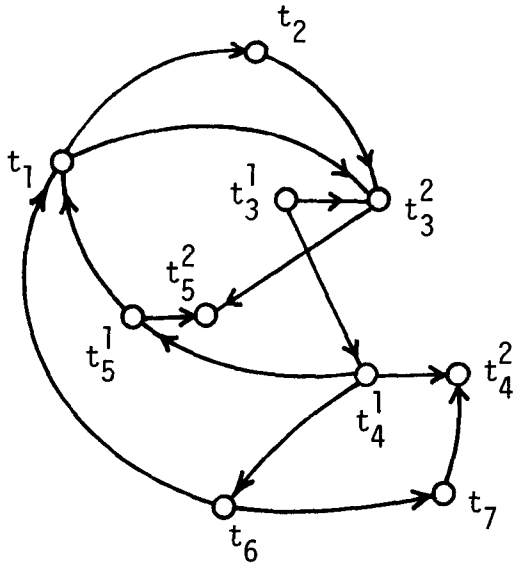


Fig. 2.11. Reduction of a compound component into a cycle-free component.

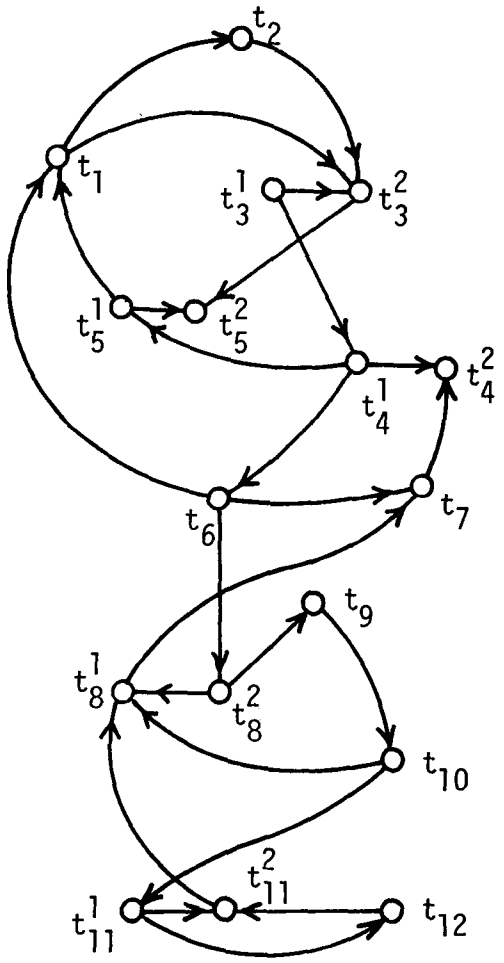


Fig. 2.12. Resulting acyclic HC-graph.

2.7 Conclusions

This chapter is characterized as the definition of the three types of the trunk-division methods. The success of the scheme developed in this thesis is entirely based upon the divided-trunk style. It is shown in Theorem 2.5 that the style can achieve one-hundred percent wirability. In this sense the approach of this thesis that restricts trunk-division methods to only three types may be said to be valid. There still remains the problem of how to realize a given set of nets in the minimum possible width, which is discussed in later chapters.

CHAPTER 3

CYCLE ELIMINATING PROCESS

3.1 Introduction

The previous chapter concentrates on the problem whether a given set of nets is realizable by using the three methods and gives no attention to the width of the resulting layout pattern. This chapter analyzes the effects of dividing trunks over the width of the wiring area. Some concepts are introduced for this purpose. Also, the chapter aims to extend the applicable range of the methods I and I', and considers a pair of cycles for which any dividing patterns do not satisfy the conditions of Theorem 2.2, but which can be eliminated without producing any new cycles if two trunks are simultaneously divided.

The section 3.3 presents an algorithm for eliminating all of cyclic constraints in a given HC-graph with minimum increase of the trunk-crossing count.

3.2 Outline of the Whole Algorithm

This section describes the outline of the wire routing algorithm presented in this and the following chapters. The algorithm consists of three functional blocks. The author calls them Algorithm A, Algorithm B and Algorithm C, respectively. Each of them is based upon a heuristic search method.

Outline of the whole algorithm:

M1: (input) Give a set of nets.

M2: Represent each net by a trunk. Let R be a set of trunks. Then, construct an HC-graph.

M3: If the HC-graph contains no cycle then go to M5.

M4: Eliminate all of cycles in the HC-graph (Algorithm A).

M5: Lay out the set of trunks R in the minimum possible width (Algorithm C).

M6: If $W(R) = W_0(R)$ then stop.

M7: Divide some trunk in a suitable way so that the track count $W(R)$ be reduced, and then return to M5 (Algorithm B).

END;

3.3 Horizontal Ordering and Trunk Groups

This section introduces some basic quantities which may be considered to be titely associated with the second criterion mentioned earlier.

Definition 3.1 Let R be a set of trunks. Let $n(y_i)$ denote the number of trunks which cross the line $y = y_i$. The maximum value of $\{n(y_i)\}$ is denoted as $W_0(R)$. If $n(y_i) = W_0(R)$ then the point y_i is said to be critical.

(Remark) Consider a trunk t_i such that $U(t_i) = L(t_i)$. For such a trunk t_i it must be true that $\#T_L(t_i) = \#T_R(t_i) = 1$ and $T_L(t_i) = T_R(t_i)$. In other words, there exists no vertical line segment corresponding to t_i. Such a net, which can always be wired without any influence on the wiring width, can be removed from a set of nets. In this and the following chapters it is assumed that a set of nets does not contain such nets.

Width of a wiring area is determined by the number of tracks to be prepared in the area on which trunks run. In general, it may be required to place several trunks on one track.

Definition 3.2 Let R be a set of trunks. The track count $W(R)$ is the minimum possible number of tracks for arranging the trunks.

(Remark) $W_0(R)$ and $W(R)$ are referred to as the trunk-crossing count

and the track count, respectively.

Definition 3.3 Horizontal-ordering functions from the left side and from the right side, denoted as $\hat{x}_L$ and $\hat{x}_R$, respectively, are defined as follows.

$$\hat{x}_L(t_i) = \begin{cases} 1 & \text{if } \Gamma^{-1}t_i = \emptyset, \\ \max\{\hat{x}_L(t_j) + 1 \mid t_j \in \Gamma^{-1}t_i\} & \text{if } \Gamma^{-1}t_i \neq \emptyset, \end{cases}$$

and

$$\hat{x}_R(t_i) = \begin{cases} 1 & \text{if } \Gamma t_i = \emptyset, \\ \max\{\hat{x}_R(t_j) + 1 \mid t_j \in \Gamma t_i\} & \text{if } \Gamma t_i \neq \emptyset. \end{cases}$$

Property 3.1 Horizontal-ordering functions $\hat{x}_L$ and $\hat{x}_R$ are defined for all trunks in R if and only if the HC-graph is acyclic.

(Proof omitted)

Property 3.2 For any set of trunks R,

$$W(R) \geq \max\{\hat{x}_L(t_i) + \hat{x}_R(t_i) - 1 \mid t_j \in R\}.$$

(Proof omitted)

Property 3.3 Let $x(t_i)$ be the x-coordinate of the track on which the trunk t_i can be placed. Then,

$$\hat{x}_L(t_i) \leq x(t_i) \leq W(R) - \hat{x}_R(t_i) + 1.$$

(Proof omitted)

The discussions so far concentrate on the horizontal constraint aspect. Vertical relations between trunks are represented by trunk groups defined below.

Definition 3.4 Trunk groups are sets of trunks G_1, G_2, ... , obtained from the following algorithm.

[Algorithm for determining trunk groups]

```
begin  y_0 := min{U(t_i) | t_i ε R};
       y_∞ := max{L(t_i) | t_i ε R};
       k := 1;  G_1 := ∅;
```

```
for y := y_0, y + 1 while y ≤ y_∞ do
    begin A := { t_i | U(t_i) ≤ y ≤ L(t_i) };
          if G_k ⊆ A then G_k := A
          else if G_k ⊄ A then
              begin k := k + 1; G_k := A end
    end
end;
```

The above algorithm is used only for the purpose of determining trunk groups. Trunk groups themselves can be obtained in a more efficient way mentioned later.

Max groups are trunk groups with the maximum cardinality. Evidently, for a max group G_k, $\#G_k = W_0(R)$. This means that if minimum-width wiring should be possible then the layout pattern must be one such that each track contains one by one from every max group.

Combination of trunk groups and horizontal-ordering functions leads to more accurate horizontal-ordering functions defined below.

Definition 3.5 For a trunk t_i, $x_L(t_i)$ and $x_R(t_i)$ are defined as follows:

$$x_L(t_i) = \begin{cases} 1 & \text{if } \hat{\Gamma}^{-1}t_i = \emptyset, \\ \max\limits_{G_j} \left[\max\limits_{t_k \in G_j \cap \hat{\Gamma}^{-1}t_i} \left(x_L(t_k) + \#\Delta_L(t_k, G_j, t_i) \right) \right] & \text{if } \hat{\Gamma}^{-1}t_i \neq \emptyset, \end{cases}$$

and

$$x_R(t_i) = \begin{cases} 1 & \text{if } \hat{\Gamma}t_i = \emptyset \\ \max\limits_{G_j} \left[\max\limits_{t_k \in G_j \cap \hat{\Gamma}t_i} \left(x_R(t_k) + \#\Delta_R(t_k, G_j, t_i) \right) \right] & \text{if } \hat{\Gamma}t_i \neq \emptyset, \end{cases}$$

where

$$\Delta_L(t_k, G_j, t_i) = \{\ t_p \mid t_p \in G_j \cap \hat{\Gamma}^{-1}t_i \text{ and } x_L(t_p) \geq x_L(t_k)\ \},$$

and

$$\Delta_R(t_k, G_j, t_i) = \{\ t_q \mid t_q \in G_j \cap \hat{\Gamma}t_i \text{ and } x_R(t_q) \geq x_R(t_k)\ \}.$$

As is evident from the above definition, x_R is the dual function of x_L. In the following it is considered what is meant by $x_L(t_i)$. Throughout the following discussion a trunk group G_j is fixed. The set $G_j \cap \hat{\Gamma}^{-1}t_i$ is the set of those trunks which belong to G_j and which must be placed to the left of t_i. Within the extent of the set $G_j \cap \hat{\Gamma}^{-1}t_i$, the set of the trunks which must be laid to the right of the track $x = x_L(t_k)$ and to the left of t_i is $\Delta_L(t_k, G_j, t_i)$. Here note that $\Delta_L(t_k, G_j, t_i) \subsetneq G_j$. Hence, the number of tracks to be prepared to the left of t_i is not less than $x_L(t_k) + \#\Delta_L(t_k, G_j, t_i)$.

The horizontal-ordering functions x_L and x_R also satisfy the properties 3.1, 3.2 and 3.3. In particular, for every trunk t_i, $x_L(t_i)$ and $x_R(t_i)$ are greater than or equal to $\hat{x}_L(t_i)$ and $\hat{x}_R(t_i)$, respectively. This means that x_L and x_R estimate the x-coordinate $x(t_i)$ of the trunk t_i more accurately than $\hat{x}_L$ and $\hat{x}_R$.

3.4 Algorithm A

--- Algorithm for eliminating all of cycles in an HC-graph ---

Algorithm A eliminates all of cyclic constraints in a given HC-graph with minimum possible increase of the trunk-crossing count. When a trunk is divided in order to eliminate cycles, the following three are required:

(1) Minimize the number of trunks to be divided.

(2) Minimize the increase of the trunk-crossing count.

(3) Minimize the following value when all cycles are eliminated;

$$\max\{\ x_L(t_i) + x_R(t_i) - 1 \mid t_i \in R\ \}.$$

These requirements are reflected on the heuristic function F_A used in the algorithm. The function F_A is defined for every combination (t_i, y_D, M) of a trunk t_i to be divided, a dividing point y_D and a dividing method M, as follows. Here, the set of trunks after dividing a trunk t_i at $y = y_D$ by a method M is denoted as $\mathcal{R}(t_i, y_D, M)$ or simply as $\mathcal{R}'$.

(1) $F_A(t_i, y_D, M) = \omega$ (undefined) if it is impossible to divide the trunk t_i at $y = y_D$ by the method M.

(2) $F_A(t_i, y_D, M) = \omega$ if the division generates any new cycles.

(3) $F_A(t_i, y_D, M) = \omega$ if $W_0(\ (t_i, y_D, M)) > W_0$, where W_0 is determined in the algorithm.

(4) $F_A(t_i, y_D, M) = \omega$ if there is no cycle containing t_i.

(5) Otherwise, $F_A(t_i, y_D, M) = \Sigma_{i=1}^{4}\ c_i \cdot f_i(t_i, y_D, M)$, where $c_1 \sim c_4$ are positive constants and $f_1 \sim f_4$ are defined as follows:

$f_1(t_i, y_D, M)$ = (the number of cycles in the HC-graph $G_h(\mathcal{R}', \Gamma)$).

$$f_2(t_i, y_D, M) = \begin{cases} 2\times\#\mathcal{R} & \text{if the HC-graph } G_h(\mathcal{R}', \Gamma) \text{ contains cycles,} \\ \max\{\ x_L(t_j) + x_R(t_j) - 1 \mid t_j \in \mathcal{R}'\ \} & \text{otherwise.} \end{cases}$$

$f_3(t_i, y_D, M)$ = (the number of terminals) - (the number of those points for the set $\mathcal{R}'$ for which $n(y) \geq W_0(\mathcal{R})$).

$f_4(t_i, y_D, M)$ is determined by the number of arcs generated by the division.

$f_4(t_i, y_D, M) = 0$ if i) neither $t_L(y_D)$ nor $t_R(y_D)$ is defined, or if ii) $y_D \in T_L(t_i) \cup T_R(t_i)$.

$f_4(t_i, y_D, M) = 1$ if i) $t_L(y_D)$ only is defined and $\hat{\Gamma}t_i \not\ni t_L(y_D)$, or if ii) $t_R(y_D)$ only is defined and $\hat{\Gamma}^{-1}t_i \not\ni t_R(y_D)$.

$f_4(t_i, y_D, M) = 2$ if both $t_L(y_D)$ and $t_R(y_D)$ are defined and both differ from the trunk t_i.

In the above, $t_L(y_D)$ and $t_R(y_D)$ represent the trunks which connect

to the left and right terminals at $y = y_D$, respectively. If none of the conditions is satisfied, new cycles are produced and hence $F_A(t_i, y_D, M)$ is undefined. If t_i is divided by the method III or III', $f_4(t_i, y_D, M) = 0$ since $y_D \in T_R(t_i)$ or $y_D \in T_L(t_i)$, respectively.

[Algorithm A]

A1: If the HC-graph $G_h(R, \Gamma)$ contains no cycle, then stop.

A2: $W_0 := W_0(R)$.

A3: Compute $F_A(t_i, y_D, M)$ for all dividing patterns.

A4: If $F_A(t_i, y_D, M)$ is undefined for all dividing patterns then go to A7.

A5: Choose a dividing pattern (t_i, y_D, M) to give the minimum value of F_A, and divide the trunk t_i at $y = y_D$ by the method M.

A6: Go to A1 with $R := R(t_i, y_D, M)$.

A7: If $W_0 > W_0(R)$ then stop, else go to A3 with $W_0 := W_0 + 1$.

End Algorithm A;

The performance of Algorithm A is outlined below. First of all, cycles are searched in a given HC-graph. If no cycle is found then it terminates since no more process is needed. If any cycles are found then it finds a set of all those dividing patterns which can eliminate a cycle(s) without any increase of the trunk-crossing count $W_0(R)$. This is achieved by setting $W_0 := W_0(R)$ at the step A2. If the above-mentioned set of dividing patterns is not null then one of them is chosen so that the most cycles are to be eliminated (estimation by f_1). On the other hand, if that set is null then the similar process follows with $W_0 := W_0(R) + 1$, which means that those dividing patterns are admitted which increase the trunk-crossing count. When it gets to the stage where all cycles can be eliminated, the algorithm, in this turn, chooses a dividing pattern to give the minimum value of $\max\{ x_L(t_j) + x_R(t_j) + 1 \mid t_j \in R' \}$

(estimation by f_2).

Algorithm A has two exits A1 and A7. If it terminates at the step A7, then there still remain cycles in the HC-graph. For example, when the set of trunks characterized in Theorem 2.5 is inputted, it terminates at the step A7 without going through the step A5. In this case an additional procedure is needed in order to guarantee perfect wireability. The first step is to find a trunk t_j such that $T_L(t_j) = T_R(t_j)$ and $\#T_L(t_j) \geq 2$. Second, divide the trunk t_j by the method III. This process creates a pair of terminals at the same column, analogous to that of empty terminals. Then the control returns to the step A2 of Algorithm A. If there exists no such trunk t_j, then divide an appropriate trunk at $y = U - 1$ or at $y = L + 1$ by the method II where U and L represent the y-coordinates of the upper and lower bounds of the wiring area, respectively, and then the control returns to A2. This additional procedure is rarely demanded for actual sets of nets, so it may be said that the procedure is needed only for theoretical assurance of perfect wireability.

The step A3 of Algorithm A takes the most time in the algorithm. At the step, F_A values must be computed for all dividing patterns. The following discusses the way to reduce the time necessary for computing $F_A(t_i, y_D, M)$.

(1) Computation of $f_1(t_i, y_D, M)$

Given a dividing pattern (t_i, y_D, M), $f_1(t_i, y_D, M)$ is computed only when this division generates no new cycle. Hence, $f_1(t_i, y_D, M)$ is equal to $N_C(R)$ minus the number of cycles eliminated by this division, where $N_C(R)$ is the number of all cycles in the HC-graph.

The followings are the equations for computing $f_1(t_i, y_D, M)$.

For each cycle C_j, let $\Gamma t_i \cap C_j = \{t_{out,j}\}$ and $\Gamma^{-1} t_i \cap C_j = \{t_{in,j}\}$.

$$f_1(t_i, y_D, \mathrm{I}) = N_C(\mathcal{R}) - \#\{\, C_j \mid C_j \ni t_i \text{ and } \min R(t_{in,j}, t_i) > y_D > \max R(t_i, t_{out,j}) \,\},$$

$$f_1(t_i, y_D, \mathrm{I}') = N_C(\mathcal{R}) - \#\{\, C_j \mid C_j \ni t_i \text{ and } \min R(t_i, t_{out,j}) > y_D > \max R(t_{in,j}, t_i) \,\},$$

$$f_1(t_i, y_D, \mathrm{II}) = N_C(\mathcal{R}) - \#\{\, C_j \mid C_j \ni t_i \,\},$$

$$f_1(t_i, y_D, \mathrm{III}) = N_C(\mathcal{R}) - \#\{\, C_j \mid C_j \ni t_i \text{ and } T_L(t_{in,j}) \not\ni y_D \,\},$$

$$f_1(t_i, y_D, \mathrm{III}') = N_C(\mathcal{R}) - \#\{\, C_j \mid C_j \ni t_i \text{ and } T_R(t_{out,j}) \not\ni y_D \,\}.$$

(2) Computation of $f_2(t_i, y_D, M)$

Given a dividing pattern (t_i, y_D, M), $f_2(t_i, y_D, M)$ is computed only when the division eliminates all of cycles in the HC-graph. Here it should be noted that if there exist dividing patterns which eliminate all of cycles then one of them is always chosen at the step A5 because of the coefficient of $\#\mathcal{R}$ in the definition of f_2. In this case only it is needed to divide the trunk t_i practically and then to compute the f_2 value in the resulting HC-graph.

On the other hand the approximate value $\hat{f}_2$ of f_2 can be easily computed. $\hat{f}_2(t_i, y_D, M)$ is determined as follows:

$$\hat{f}_2(t_i, y_D, M) = \max\{\, \hat{x}_L(t_j) + \hat{x}_R(t_j) - 1 \mid t_j \in \mathcal{R}(t_i, y_D, M) \,\}.$$

Note that $\hat{x}_L(t_i)$ corresponds to the length of the longest constraint chain that terminates at t_i in the HC-graph. Hence, $\hat{f}_2(t_i, y_D, M)$ is computed by finding the longest constraint chain in the HC-graph $G_h(\mathcal{R}', \Gamma)$.

3.5 Extension of the Applicable Range of the Methods I and I'

In Algorithm A presented in the previous section, those dividing patterns are preferred that cause no increase of the trunk-crossing count. From such a standpoint the methods I and I' are used most frequently. This section investigates the extent to which cycles can be eliminated using only type I trunk divisions.

Theorem 2.2 presented in Chapter 2 concentrates on one cycle to give the condition for cycle to be eliminated by dividing a trunk without generating any new cycle. The following discussion aims to extend the applicable range of the methods I and I'.

The cases where the condition (v) of Theorem 2.2 is not satisfied are considered. They are classified into following three types:

(1) Forward type:

$$t_L(y_D) \in \hat{\Gamma} t_i, \quad t_R(y_D) \notin \hat{\Gamma}^{-1} t_i, \quad \text{and } t_L(y_D) \notin \hat{\Gamma} t_R(y_D).$$

(2) Backward type:

$$t_L(y_D) \notin \hat{\Gamma} t_i, \quad t_R(y_D) \in \hat{\Gamma}^{-1} t_i, \quad \text{and } t_L(y_D) \notin \hat{\Gamma} t_R(y_D).$$

(3) Cyclic type:

$$t_L(y_D) \notin \hat{\Gamma} t_i, \quad t_R(y_D) \notin \hat{\Gamma}^{-1} t_i, \quad \text{and } t_L(y_D) \in \hat{\Gamma} t_R(y_D).$$

In all the above cases it is assumed that $t_L(y_D)$ differs from $t_R(y_D)$.

Since the backward type can be considered as a dual situation to the forward type, any assertion which is dual to that of the forward type applies to the backward type. Consequently, no discussion is given for the backward type in the following. Fig. 3.1 illustrates the three types.

In order to make the condition (v) of Theorem 2.2 valid in the forward type, another trunk must be divided so that all of directed paths (horizontal constraint chains) from t_i to $t_L(y_D)$ are eliminated. The following considers only the case in which a division of one trunk is

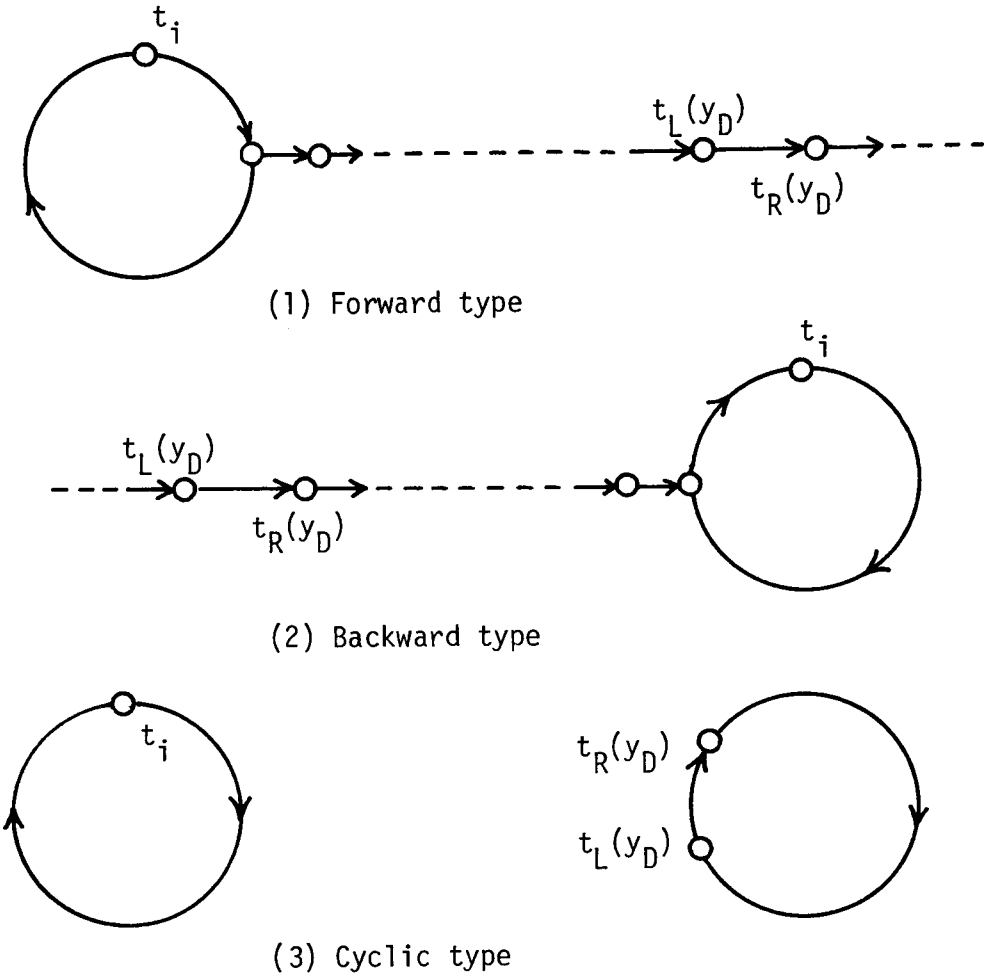


Fig. 3.1. Three cases in which the condition in Theorem 2.2 is not satisfied.

enough to cut the path from t_i to $t_L(y_D)$.

Definition 3.6 In an HC-graph G_h, let $P(t_i, t_j)$ be the set of vertices (trunks) on a directed path (or paths) from t_i to t_j. Formally, $P(t_i, t_j)$ is defined by

$$P(t_i, t_j) = \begin{cases} \emptyset & \text{if } t_j \notin \hat{\Gamma} t_i, \\ (\hat{\Gamma} t_i \cup \hat{\Gamma}^{-1} t_j) \cup \{t_i, t_j\} & \text{if } t_j \in \hat{\Gamma} t_i. \end{cases}$$

Definition 3.7 In an HC-graph G_h, let $t_j \in \hat{\Gamma} t_i$. A vertex (other than

t_i or t_j) which is contained in all of directed paths from t_i to t_j is called an articulation point of $P(t_i, t_j)$. The set of articulation points of $P(t_i, t_j)$ is denoted by $P_a(t_i, t_j)$.

In a forward-type HC-graph, directed paths from t_i to $t_L(y_D)$ are called foward paths. The following two situations are considered concerning such a foward path:

(i) The forward path contains a trunk t_m belonging to a cycle C' which does not have a common portion with the cycle C (refer to Theorem 3.1).

(ii) The forward path contains a trunk t_m other than $t_L(y_D)$ which is not contained in any cycles (refer to Theorem 3.2).

In order to simplify the discussion the following consideration assumes that no terminal is empty, i.e., every terminal is associated with some net.

Theorem 3.1 Let C and C' be two cycles in an HC-graph which are mutually disjoint. For a trunk t_i on the cycle C and a dividing point y_D, let the HC-graph be of the forward type in which

$$t_L(y_D) \in \hat{\Gamma} t_i,\ t_R(y_D) \notin \hat{\Gamma}^{-1} t_i,\ t_L(y_D) \notin \hat{\Gamma} t_R(y_D),\ \text{and}\ t_L(y_D) \neq t_R(y_D),$$

and let t_m be an articulation point of $P(t_i, t_L(y_D))$ which is on the cycle C' (the condition so far is referred to as (PC-3.1)).

When the trunk t_m is divided at $y = y_d$ by the method I or I', the necessary and sufficient condition for both the cycles C and C' to be eliminated simultaneously without producing any new cycle, is that one of the following conditions (A) and (B) is valid:

(A) Both t_i and t_m are articulation points of $P(t_R(y_d), t_L(y_D))$, where $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$ and $t_L(y_d) \neq t_R(y_d)$ are satisfied, and one of the following (1), (1') and one of (2), (2') are valid:

(1) $\min R(T_1, t_i) > y_D > \max R(t_i, T_2)$,

(1') $\min R(t_i, T_2) > y_D > \max R(T_1, t_i)$,

(2) $\min R(T_3, t_m) > y_d > \max R(t_m, T_4)$,

(2') $\min R(t_m, T_4) > y_d > \max R(T_3, t_m)$,

$$T_1 = [P(t_R(y_d), t_L(y_D)) \cup C] \cap \Gamma^{-1}t_i,$$

$$T_2 = [P(t_R(y_d), t_L(y_D)) \cup C] \cap \Gamma t_i,$$

$$T_3 = [P(t_R(y_d), t_L(y_D)) \cup C'] \cap \Gamma^{-1}t_m,$$

$$T_4 = [P(t_R(y_d), t_L(y_D)) \cup C'] \cap \Gamma t_m.$$

(B) $t_L(y_d) \in \hat{\Gamma} t_i$, $t_R(y_d) \notin \hat{\Gamma}^{-1} t_m$, $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$, $t_L(y_d) \neq t_R(y_d)$ and one of the following (3), (3') and one of (4), (4') are valid:

(3) $\min R(T_5, t_i) > y_D > \max R(t_i, T_6)$,

(3') $\min R(t_i, T_6) > y_D > \max R(T_5, t_i)$,

(4) $\min R(T_7, t_m) > y_d > \max R(t_m, T_8)$,

(4') $\min R(t_m, T_8) > y_d > \max R(T_7, t_m)$,

$$T_5 = C \cap \Gamma^{-1}t_i, \qquad T_6 = C \cap \Gamma t_i,$$

$$T_7 = [P(t_i, t_L(y_D)) \cup C'] \cap \Gamma^{-1}t_m,$$

$$T_8 = [P(t_i, t_L(y_D)) \cup C'] \cap \Gamma t_m.$$

(Proof) The trunks t_i and t_m must be divided by the method I or by the method I', according to whether the unprimed conditions are satisfied or the primed ones are satisfied. In order to simplify the proof, it is shown that the divisions of t_i and t_m by the method I eliminate the cycles C and C' without producing any new cycle if and only if the unprimed conditions are satisfied.

Consider an example to clarify the proof. Fig. 3.2 shows the HC-graph for the set of trunks which is given in Table 3.1. The HC-graph

t_i	$T_L(t_i)$	$T_R(t_i)$	$U(t_i)$	$L(t_i)$
t_1	{11}	{3}	3	11
t_2	{6, 5}	{8, 11}	5	11
t_3	{13, 16}	{6}	6	16
t_4	{3}	{13, 10}	3	13
t_5	{10, 12}	∅	10	12
t_6	{8}	{12}	8	12
t_7	{15}	{5}	5	15
t_8	{14}	{4, 16}	4	16
t_9	{9}	{14, 15}	9	15
t_{10}	{1, 7}	{9}	1	9
t_{11}	{2, 4}	{1}	1	4
t_{12}	∅	{2, 7}	2	7

Table 3.1. A set of nets $R_{3,1}$.

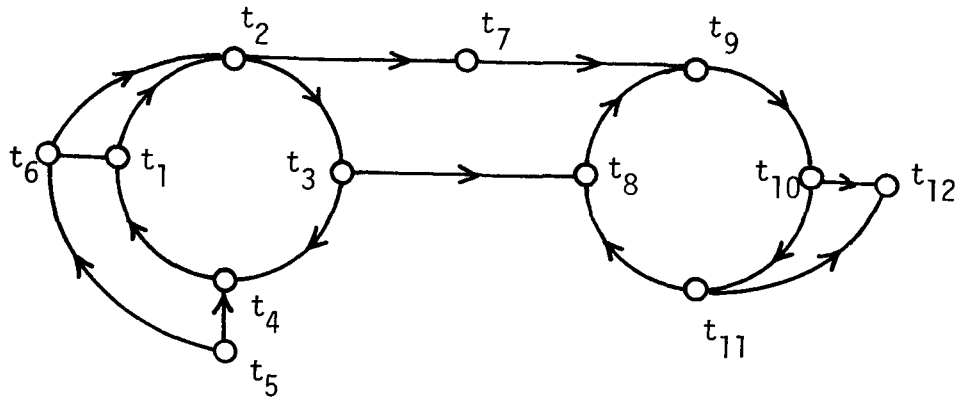


Fig. 3.2. HC-graph for the set of trunks $R_{3,1}$.

contains two cycles $C_1 = \{t_1, t_2, t_3, t_4\}$ and $C_2 = \{t_8, t_9, t_{10}, t_{11}\}$. Suppose that the trunk t_2 is to be divided at y = 7 by the method I. Then, the trunks $t_L(7) = t_{10}$ and $t_R(7) = t_{12}$ satisfy

$$t_L(7) \in \hat{\Gamma} t_2,\ t_R(7) \notin \hat{\Gamma}^{-1} t_2,\ t_L(7) \notin \hat{\Gamma} t_R(7), \text{ and } t_L(7) \neq t_R(7),$$

and the situation proves to be of the forward type. Here note that the forward paths from t_2 to $t_L(7) = t_{10}$ contain only one articulation point t_9. Divide the trunk t_9 at y = 10 by the method I, and the trunks $t_L(10) = t_5$ and $t_R(10) = t_4$ satisfy

$$P_a(t_4, t_{10}) \supseteq \{t_2, t_9\}, \text{ and}$$

$$t_L(10) \notin \hat{\Gamma} t_R(10).$$

The remaining conditions of (A) are also verified in the below:

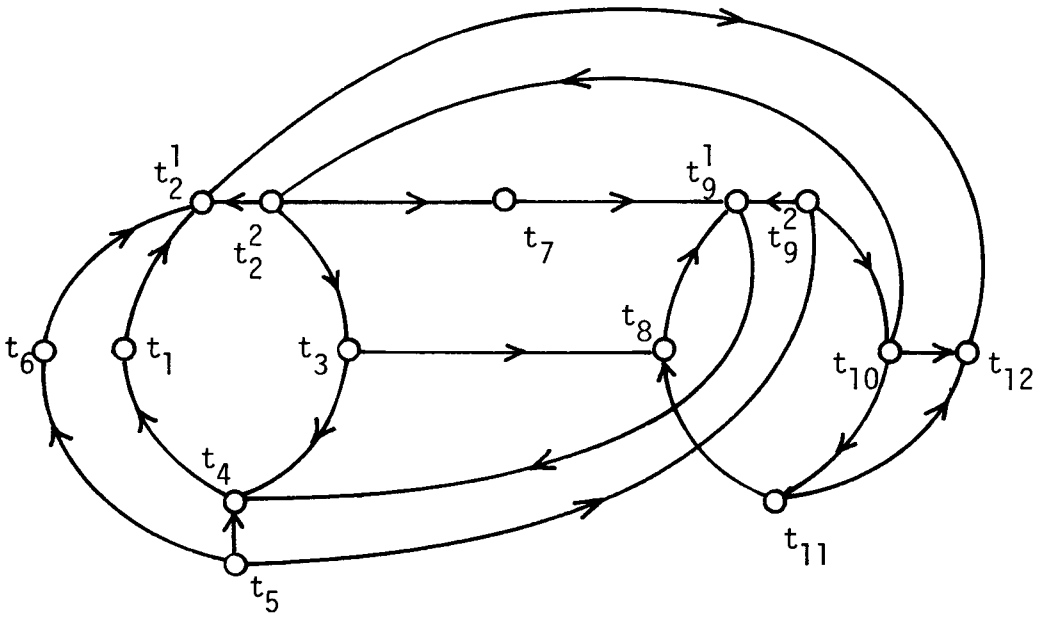


Fig. 3.3. Resulting acyclic HC-graph.

$$T_1 = [P(t_4, t_{10}) \cup C_1] \cap \Gamma^{-1}t_2 = \{ t_1 \},$$

$$T_2 = [P(t_4, t_{10}) \cup C_1] \cap \Gamma t_2 = \{t_3, t_7\},$$

$$R(T_1, t_2) = R(t_1, t_2) = \{ 11 \},$$

$$R(t_2, T_2) = R(t_2, t_3) \cup R(t_2, t_7) = \{5, 6\},$$

$$\min R(T_1, t_2) > 7 > \max R(t_2, T_2),$$

and

$$T_3 = [P(t_4, t_{10}) \cup C_2] \cap \Gamma^{-1}t_9 = \{t_7, t_8\},$$

$$T_4 = [P(t_4, t_{10}) \cup C_2] \cap \Gamma t_9 = \{ t_{10} \},$$

$$R(T_3, t_9) = R(t_7, t_9) \cup R(t_8, t_9) = \{14, 15\},$$

$$R(t_9, T_4) = R(t_9, t_{10}) = \{ 9 \},$$

$$\min R(T_3, t_9) > 10 > \max R(t_9, T_4).$$

Fig 3.3 shows the acyclic HC-graph after dividing t_2 and t_9 at y = 7 and at y = 10 respectively, both by the method I.

Before proceeding to the proof of Theorem 3.1, some definitions and lemmas are needed.

Definition 3.8 For an HC-graph $G_h = (R, E)$ and a subset A of E where E is the set of arcs of G_h, an A-reduced HC-graph G_h/A is a directed graph obtained by deleting from G_h all arcs belonging to A.

Definition 3.9 For an HC-graph $G_h = (R, E)$, let $G_h(T_d)$ be the HC-graph resulting from dividing trunks belonging to the set T_d by the method I. Also, let $A(T_d)$ be the set of those arcs which are newly created by divisions of these trunks. Then, the $A(T_d)$-reduced HC-graph $G_h(T_d)/A(T_d)$ is the graph obtained by deleting all of those newly created arcs from the HC-graph after dividing the trunks of T_d.

Definition 3.10 For a set of trunks T_d, let $d(t_i) = \{t_i^1, t_i^2\}$ if $t_i \in T_d$ where t_i^1 and t_i^2 are trunks resulting from dividing t_i by the method I,

and let $d(t_i) = \{ t_i \}$ if $t_i \notin T_d$.

Lemma 3.1 For an HC-graph G_h and any set of trunks T_d, if $t_j \notin \hat{\Gamma} t_i$ in G_h then $(\hat{\Gamma} d(t_i)) \cap d(t_j) = \emptyset$ in the $A(T_d)$-reduced HC-graph $G_h(T_d)/A(T_d)$.

(Proof omitted)

Definition 3.11 For an HC-graph $G_h(R, E)$ and a set of trunks T_d, the connection graph $G^* = (A(T_d), E_A)$ for T_d is defined as follows:

(1) The vertex set of G^* corresponds to the set $A(T_d)$ of the arcs which are newly created by dividing the trunks of T_d.

(2) Let e_i and e_j be two arcs in $A(T_d)$. Then, in G^*, a directed arc is drawn from the vertex e_i to the vertex e_j in G^* if and only if there exists any directed path from the final point of e_i to the starting point of e_j in the $A(T_d)$-reduced HC-graph $G_h(T_d)/A(T_d)$.

Lemma 3.3 Let $G_h(R, E)$ be an HC-graph and T_d be a subset of R. Then, divisions of the trunks belonging to T_d by the method I produce no new cycles if and only if the connection graph $G^* = (A(T_d), E_A)$ does not contain any cycles or self-loops.

(Proof omitted)

For a connection graph G^*, the incidence matrix $P = (p_{ij})$ is defined by

$$p_{ij} = \begin{cases} 1 & \text{if } (e_i, e_j) \in E_A, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 3.3 The connection graph $G^* = (A(T_d), E_A)$ does not contain any cycles or any self-loops if and only if the incidence matrix P for G^* satisfies

$$\mathrm{Per}(P + I) = 1,$$

where I is the unit matrix and Per represents the permanent expansion of a matrix [22]. (Proof omitted)

Now continue with the proof of Theorem 3.1.

Let

$$e_1 = (t_i^1,\ t_R(y_D)), \qquad e_2 = (t_L(y_D),\ t_i^2),$$

$$e_3 = (t_m^1,\ t_R(y_d)), \qquad e_4 = (t_L(y_d),\ t_m^2),$$

$$T_d = \{t_i,\ t_m\}, \qquad A(T_d) = \{e_1,\ e_2,\ e_3,\ e_4\}.$$

Then it suffices to show that the unprimed condition of the theorem is the necessary and sufficient condition that the cycles C and C' are eliminated and that the connection graph $G^* = (A(T_d),\ E_A)$ does not contain any cycles or any self-loops.

The necessary and sufficient condition for the cycles C and C' to be eliminated by dividing t_i and t_m, is that

$$\min R(T_1',\ t_i) > y_D > \max R(t_i,\ T_2'),$$

and

$$\min R(T_3',\ t_m) > y_d > \max R(t_m,\ T_4'),$$

where

$$T_1' = C \cap \Gamma^{-1}t_i, \qquad T_2' = C \cap \Gamma t_i,$$

$$T_3' = C' \cap \Gamma^{-1}t_m, \qquad T_4' = C' \cap \Gamma t_m.$$

The necessary and sufficient condition for the connection graph $G^* = (A(T_d),\ E_A)$ not to contain a cycle or a self-loop is, from Lemma 3.3, that $\mathrm{Per}(P + I) = 1$ holds for the incidence matrix P for G^*.

Note the incidence matrix P. Since obviously $t_i^1 \notin \hat{\Gamma}t_R(y_D)$, $t_L(y_D) \notin \hat{\Gamma}t_R(y_D)$, $t_i^1 \in \hat{\Gamma}t_i^2$, and $t_m^1 \in \hat{\Gamma}t_m^2$ hold in $G_h(T_d)/A(T_d)$, it follows that $p_{11} = p_{12} = 0$ and $p_{21} = p_{43} = 1$. It is shown below that $p_{13} = p_{41} = 0$ and $p_{23} = p_{42} = 1$ follow from the condition (PC-3.1).

(1) Proof of $p_{13} = 0$ ($t_m^1 \notin \hat{\Gamma}t_R(y_D)$ in $G_h(T_d)/A(T_d)$).

Assume that $t_m \varepsilon \hat{\Gamma} t_R(y_D)$ in G_h, and we can see that $t_L(y_D) \varepsilon \hat{\Gamma} t_m$, from the condition (PC-3.1), and thus $t_L(y_D) \varepsilon \hat{\Gamma} t_R(y_D)$ in G_h, which is a contradiction. It follows that $t_m \notin \hat{\Gamma} t_R(y_D)$ holds in G_h. Consequently, it must be valid from Lemma 3.1 that $t_m^1 \notin \hat{\Gamma} t_R(y_D)$, i.e., $p_{13} = 0$.

(2) Proof of $p_{41} = 0$ ($t_i^1 \notin \hat{\Gamma} t_m^2$ in $G_h(T_d)/A(T_d)$).

Since $t_i \notin \hat{\Gamma} t_m$ follows from the condition (PC-3.1), $t_i^1 \notin \hat{\Gamma} t_m^2$ in $G_h(T_d)/A(T_d)$ follows from Lemma 3.1.

(3) Proof of $p_{23} = 1$ ($t_m^1 \varepsilon \hat{\Gamma} t_i^2$ in $G_h(T_d)/A(T_d)$).

Since $t_m \varepsilon \hat{\Gamma} t_i$ in G_h and $t_m^1 \varepsilon \hat{\Gamma} t_m^2$ and $t_i^1 \varepsilon \hat{\Gamma} t_i^2$ in $G_h(T_d)/A(T_d)$, it follows that $t_m^1 \varepsilon \hat{\Gamma} t_i^2$.

(4) Proof of $p_{42} = 1$ ($t_L(y_D) \varepsilon \hat{\Gamma} t_m^2$ in $G_h(T_d)/A(T_d)$).

Since $t_L(y_D) \varepsilon \hat{\Gamma} t_m$ in G_h and $t_m^1 \varepsilon \hat{\Gamma} t_m^2$ in $G_h(T_d)/A(T_d)$, it follows that $t_L(y_D) \varepsilon \hat{\Gamma} t_m^2$.

Consequently, the incidence matrix P is of the following form:

$$P + I = \begin{bmatrix} 1, & 0, & 0, & p_{14} \\ 1, & 1 + p_{22}, & 1, & p_{24} \\ p_{31}, & p_{32}, & 1 + p_{33}, & p_{34} \\ 0, & 1, & 1, & 1 + p_{44} \end{bmatrix}$$

Calculating Per(P + I),

$$\begin{aligned} \mathrm{Per}(P + I) = {} & (1 + p_{22})(1 + p_{33})(1 + p_{44}) + p_{32}(1 + p_{44} + p_{24}) \\ & + p_{34}(2 + p_{22}) + p_{24}(1 + p_{33}) \\ & + p_{14}[1 + p_{32} + p_{33} + p_{31}(2 + p_{22})]. \end{aligned}$$

It follows that the necessary and sufficient condition for Per(P + I)

= 1 can be expressed as follows:

(5) Requirement for Per(P + I) = 1:

$$p_{22} = p_{33} = p_{44} = p_{14} = p_{24} = p_{34} = p_{32} = 0.$$

In the following, it is shown that the condition for (5) to hold is the condition given in the theorem. The case then divides into that of $t_m \in \hat{\Gamma} t_R(y_d)$ and that of $t_m \notin \hat{\Gamma} t_R(y_d)$.

(A) The case of $t_m \in \hat{\Gamma} t_R(y_d)$.

In order for $p_{33} = 0$ to hold, i.e., $t_m^1 \notin \hat{\Gamma} t_R(y_d)$ in $G_h(T_d)/A(T_d)$, the directed path from $t_R(y_d)$ to t_m must be cut. If there exists a directed path from $t_R(y_d)$ to t_m without going through t_i, then any divisions of t_i and t_m leave a directed path from $t_R(y_d)$ to t_m^1 in $G_h(T_d)/A(T_d)$ since $t_m^1 \in \hat{\Gamma} t_m^2$.

Consequently, $p_{33} = 0$ requires that the vertex t_i is an articulation point of $P(t_R(y_d), t_m)$, that is,

(6) $P_a(t_R(y_d), t_m) \ni t_i$.

Then the condition for all of the directed paths from $t_R(y_d)$ to t_m to be cut is that

$$\min R(T_1'', t_i) > y_D > \max R(t_i, T_2''),$$

where

$$T_1'' = P(t_R(y_d), t_m) \cap \Gamma^{-1} t_i,$$

$$T_2'' = P(t_R(y_d), t_i) \cap \Gamma t_i.$$

Next to consider is the condition for $p_{22} = 0$. From the condition (PC-3.1), $t_L(y_D) \in \hat{\Gamma} t_i$ and $P_a(t_i, t_L(y_D)) \ni t_m$ in G_h. In order to get the situation that $t_L(y_D) \notin \hat{\Gamma} t_i^2$ in $G_h(T_d)/A(T_d)$, it is necessary that the directed path from t_i^2 to $t_L(y_D)$ must be cut off by dividing t_m, i.e.,

$$\min R(T_3'', t_m) > y_d > \max R(t_m, T_4''),$$

where

$$T_3'' = P(t_i, t_L(y_D)) \cap \Gamma^{-1} t_m,$$

$$T_4'' = P(t_i, t_L(y_D)) \cap \Gamma t_m.$$

Now consider what conditions are further needed for $p_{32} = 0$, i.e., $t_L(y_D) \notin \hat{\Gamma} t_R(y_d)$ in $G_h(T_d)/A(T_d)$. From the condition (PC-3.1) and (6) above, it is seen that $P_a(t_i, t_L(y_D)) \ni t_m$ and $P_a(t_R(y_d), t_m) \ni t_i$. Hence, $t_L(y_D) \in \hat{\Gamma} t_R(y_d)$ in G_h. Then, only the following two cases need consideration:

(7) Both t_i and t_m are articulation points of $P(t_R(y_D), t_L(y_D))$.

(8) There exists a directed path from $t_R(y_d)$ to $t_L(y_D)$ without going through t_i or t_m.

In case (8), obviously $t_L(y_D) \in \hat{\Gamma} t_R(y_d)$ in $G_h(T_d)/A(T_d)$, and $p_{32} = 1$ follows. On the other hand in case (7), in order for the directed path from $t_R(y_d)$ to $t_L(y_D)$ to be cut in $G_h(T_d)/A(T_d)$, the following relations are necessary:

$$\min R(T_1''', t_i) > y_D > \max R(t_i, T_2''')$$

and

$$\min R(T_3''', t_m) > y_d > \max R(t_m, T_4'''),$$

where

$$T_1''' = P(t_R(y_d), t_L(y_D)) \cap \Gamma^{-1} t_i,$$

$$T_2''' = P(t_R(y_d), t_L(y_D)) \cap \Gamma t_i,$$

$$T_3''' = P(t_R(y_d), t_L(y_D)) \cap \Gamma^{-1} t_m,$$

$$T_4''' = P(t_R(y_d), t_L(y_D)) \cap \Gamma t_m.$$

Here note that $T_i''' = T_i''$ for i = 1, 2, 3, 4 since t_i and t_m are articulation points of $P(t_R(y_d), t_L(y_D))$, and also $T_i' \cup T_i'' = T_i$ for any i.

Lastly, it is shown that $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$ in G_h is sufficient for

$p_{44} = p_{14} = p_{24} = p_{34} = 0$.

(9) $p_{34} = 0$ ($t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$ in $G_h(T_d)/A(T_d)$)

Assume that $t_L(y_d) \in \hat{\Gamma} t_R(y_d)$ in G_h. Then, in order for $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$ to hold in $G_h(T_d)/A(T_d)$, $P(t_R(y_d), t_L(y_d))$ must contain t_i or t_m. If it contains t_m then $t_L(y_d) \in \hat{\Gamma} t_m$, i.e., $p_{44} = 1$, and if it contains t_i then $t_L(y_d) \in \hat{\Gamma} t_i^2$, i.e., $p_{24} = 1$, either of which contradicts other conditions. Consequently, the necessary and sufficient condition for $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$ in $G_h(T_d)/A(T_d)$ is that $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$ in G_h.

(10) $p_{24} = 0$ ($t_L(y_d) \notin \hat{\Gamma} t_i^2$ in $G_h(T_d)/A(T_d)$)

Since $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$, $t_L(y_d) \notin \hat{\Gamma} t_i$ in G_h. Hence, $t_L(y_d) \notin \hat{\Gamma} t_i^2$ in $G_h(T_d)/A(T_d)$.

(11) $p_{44} = 0$ ($t_L(y_d) \notin \hat{\Gamma} t_m^2$ in $G_h(T_d)/A(T_d)$)

Assuming that $t_L(y_d) \in \hat{\Gamma} t_m$ in G_h, it follows that $t_L(y_d) \in \hat{\Gamma} t_i$, which contradicts (10). Hence, $t_L(y_d) \notin \hat{\Gamma} t_m^2$ in $G_h(T_d)/A(T_d)$.

(12) $p_{14} = 0$ ($t_L(y_d) \notin \hat{\Gamma} t_R(y_D)$ in $G_h(T_d)/A(T_d)$)

Assuming that $t_L(y_d) \in \hat{\Gamma} t_R(y_D)$ in G_h, then $t_R(y_d) \in \hat{\Gamma} t_L(y_d)$ and $t_L(y_D) \in \hat{\Gamma} t_R(y_d)$ and thus $t_L(y_D) \in \hat{\Gamma} t_R(y_D)$, which contradicts the condition (PC-3.1). Consequently, $t_L(y_d) \notin \hat{\Gamma} t_R(y_D)$ in $G_h(T_d)/A(T_d)$.

Integration of the above conditions gives the condition (A) of the theorem.

(B) The case of $t_m \notin \hat{\Gamma} t_R(y_d)$

From Lemma 3.1, $t_m^1 \notin \hat{\Gamma} t_R(y_d)$ in $G_h(T_d)/A(T_d)$. The condition $p_{22} = 0$ is the same as in (A). The following is the discussion of the condition for $p_{32} = p_{34} = p_{24} = p_{44} = p_{14} = 0$.

(13) $p_{32} = 0$ ($t_L(y_D) \notin \hat{\Gamma} t_R(y_d)$ in $G_h(T_d)/A(T_d)$)

Since $t_i \notin \hat{\Gamma} t_R(y_d)$ and $t_m \notin \hat{\Gamma} t_R(y_d)$ in G_h, $t_L(y_D) \notin \hat{\Gamma} t_R(y_d)$ in G_h

must be the case for $t_L(y_D) \notin \hat{\Gamma} t_R(y_d)$ to hold in $G_h(T_d)/A(T_d)$.

(14) $p_{34} = 0$ ($t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$ in $G_h(T_d)/A(T_d)$)

The reasoning is similar to that in case (A).

(15) $p_{24} = 0$ ($t_L(y_d) \notin \hat{\Gamma} t_i^2$ in $G_h(T_d)/A(T_d)$)

From the reasoning in (9), $t_L(y_d) \notin \hat{\Gamma} t_i$ must be the case in G_h in order for $t_L(y_d) \notin \hat{\Gamma} t_i^2$ to hold in $G_h(T_d)/A(T_d)$.

(16) $p_{44} = 0$ ($t_L(y_d) \notin \hat{\Gamma} t_m^2$ in $G_h(T_d)/A(T_d)$)

The reasoning is similar to that in case (A).

(17) $p_{14} = 0$ ($t_L(y_d) \notin \hat{\Gamma} t_R(y_D)$ in $G_h(T_d)/A(T_d)$)

Assuming that $t_L(y_d) \in \hat{\Gamma} t_R(y_D)$ in G_h, then $t_L(y_D) \in \hat{\Gamma} t_i$ and $t_R(y_D) \in \hat{\Gamma} t_L(y_D)$ and thus $t_L(y_d) \in \hat{\Gamma} t_i$ in G_h, which contradicts (15). Consequently, $t_L(y_d) \notin \hat{\Gamma} t_R(y_D)$ in $G_h(T_d)/A(T_d)$.

Integration of the above conditions gives the condition (B) of the theorem.

Q. E. D.

Theorem 3.2 Let C be a cycle in an HC-graph. For a trunk t_i on the cycle C and a dividing point y_D, let the HC-graph be of the forward type in which

$t_L(y_D) \in \hat{\Gamma} t_i$, $t_R(y_D) \notin \hat{\Gamma}^{-1} t_i$, $t_L(y_D) \notin \hat{\Gamma} t_R(y_D)$, and $t_L(y_D) \neq t_R(y_D)$

and let t_m be an articulation point of $P(t_i, t_L(y_D))$ which is not contained in any cycle.

Then, the cycle C is eliminated without generating any new cycle by dividing the trunk t_i at $y = y_D$ and t_m at $y = y_d$ both by the method I or I' if and only if one of the following conditions (A) and (B) holds:

(A) Both t_i and t_m are articulation points of $P(t_R(y_d), t_L(y_D))$, where $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$ and $t_L(y_d) \neq t_R(y_d)$ are satisfied, and one of the

following (1), (1') and one of (2), (2') are valid:

$$(1)\quad \min R(T_1, t_i) > y_D > \max R(t_i, T_2),$$

$$(1')\quad \min R(t_i, T_2) > y_D > \max R(T_1, t_i),$$

$$(2)\quad \min R(T_3, t_m) > y_d > \max R(t_m, T_4),$$

$$(2')\quad \min R(t_m, T_4) > y_d > \max R(T_3, t_m),$$

where

$$T_1 = [P(t_R(y_d), t_L(y_D)) \cup C] \cap \Gamma^{-1}t_i,$$

$$T_2 = [P(t_R(y_d), t_L(y_D)) \cup C] \cap \Gamma t_i,$$

$$T_3 = P(t_R(y_d), t_L(y_D)) \cap \Gamma^{-1}t_m,$$

$$T_4 = P(t_R(y_d), t_L(y_D)) \cap \Gamma t_m.$$

(B) $t_L(y_d) \notin \hat{\Gamma} t_i$, $t_R(y_d) \notin \hat{\Gamma}^{-1} t_m$, $t_L(y_d) \notin \hat{\Gamma} t_R(y_d)$, $t_L(y_d) \neq t_R(y_d)$

and one of the following (3), (3') and one of (4), (4') are valid:

$$(3)\quad \min R(T_5, t_i) > y_D > \max R(t_i, T_6),$$

$$(3')\quad \min R(t_i, T_6) > y_D > \max R(T_5, t_i),$$

$$(4)\quad \min R(T_7, t_m) > y_d > \max R(t_m, T_8),$$

$$(4')\quad \min R(t_m, T_8) > y_d > \max R(T_7, t_m),$$

where

$$T_5 = C \cap \Gamma^{-1}t_i, \qquad T_6 = C \cap \Gamma t_i,$$

$$T_7 = P(t_i, t_L(y_D)) \cap \Gamma^{-1}t_m,$$

$$T_8 = P(t_i, t_L(y_D)) \cap \Gamma t_m.$$

(Proof omitted)

Theorem 3.3 Let C and C' be two cycles in an HC-graph which are mutually disjoint. For a trunk t_i on the cycle C and a dividing point y_D, let the HC-graph be of the cyclic type in which

$$t_L(y_D) \notin \hat{\Gamma} t_i,\ t_R(y_D) \notin \hat{\Gamma}^{-1} t_i,\ t_L(y_D) \in \hat{\Gamma} t_R(y_D), \text{ and } t_L(y_D) \neq t_R(y_D)$$

and let C' be the cycle which contains both $t_L(y_D)$ and $t_R(y_D)$.

Then, the cycles C and C' are eliminated without generating any new

cycle by dividing the trunk t_i at $y = y_D$ and a trunk t_m on C' at $y = y_d$ both by the method I or I' if and only if

$$P_a(t_R(y_D), t_L(y_D) \ni t_m,$$

$$t_L(y_d) \notin \hat{\Gamma} t_m,\ t_R(y_d) \notin \hat{\Gamma}^{-1} t_m,\ t_L(y_d) \notin \hat{\Gamma} t_R(y_d),\ t_L(y_d) \neq t_R(y_d),$$

$$t_L(y_d) \notin \hat{\Gamma} t_i, \text{ and } t_R(y_d) \notin \hat{\Gamma}^{-1} t_i$$

are valid, and one of the following (1), (1') and one of (2), (2') are satisfied:

$$(1)\ \min R(T_1, t_i) > y_D > \max R(t_i, T_2),$$

$$(1')\ \min R(t_i, T_2) > y_D > \max R(T_1, t_i),$$

$$(2)\ \min R(T_3, t_m) > y_d > \max R(t_m, T_4),$$

$$(2')\ \min R(t_m, T_4) > y_d > \max R(T_3, t_m),$$

where

$$T_1 = C \cap \Gamma^{-1} t_i, \qquad T_2 = C \cap \Gamma t_i,$$

$$T_3 = C' \cap \Gamma^{-1} t_m, \qquad T_4 = C' \cap \Gamma t_m.$$

(Proof omitted)

3.6 Conclusions

The first objective of this chapter has been laid on evaluating the operation of dividing a trunk from the aspect of its ability to eliminate cyclic constraints. If the purpose of the cycle eliminating algorithm was only to achieve the perfect wirability then it would have only to choose in order dividing patterns each of which eliminates the maximum number of cycles in an HC-graph. From a practical point of view, however, the cycle eliminating algorithm must cooperate with a width reduction algorithm described in the following chapter, since the first objective of the whole wire routing algorithm is to realize a given set

of nets in the minimum possible width. Considering the above-mentioned things, this chapter has presented the algorithm for eliminating all of cycles in an HC-graph with as little increase of the trunk-crossing count as possible.

CHAPTER 4

WIDTH REDUCTION PROCESS

4.1 Introduction

The previous chapter has presented the algorithm for realizing any set of nets in finite width with minimum possible increase of the trunk-crossing count. This chapter discusses a process following the cycle eliminating algorithm, so it is assumed through this chapter that the HC-graph does not contain any cycles.

The purpose of this chapter is to formalize the minimum-width wiring problem and to determine which trunks to be divided and how to divide them in order to reduce the track count (wiring width). For this purpose several notions are introduced, such as decomposition of a trunk set, incomplete subsets, a VR-graph and etc.

Traditional approaches toward this minimum-width wiring problem have no ability to reveal the effects of dividing a trunk on the wiring width in an explicit form. This thesis evaluates such effects by making use of the two directed graphs, the HC-graph which expresses the horizontal constraints between the trunks, and the VR-graph (vertical relation graph) to express the vertical relations.

4.2 Formulation of Minimum-Width Wiring Problem

For a set of trunks R, the previous chapter defined the trunk-crossing count $W_0(R)$ and the track count $W(R)$, for which it is always valid that $W_0(R) \leq W(R)$.

In case of $W_0(R) = W(R)$ no more improvement is possible, but the case of $W(R) > W_0(R)$ needs a process to reduce the track count $W(R)$. The process must find a trunk an appropriate division of which makes the

the resultant track count $W(R')$ less than the original track count $W(R)$. Here, it should be noted that the division may also cause the increase of the trunk-crossing count, which prohibits further improvements. Accordingly, those divisions are desirable that leave the trunk-crossing count as it is, and they are called "safe" divisions. The width reduction process is iterated until no more improvement is possible.

As mentioned in the previous chapter, the wiring width $W(R)$ is determined as the number of tracks on which trunks to be placed. In general, each track should contain as many trunks as possible. This means that the trunk set R should be decomposed into mutually disjoint subsets under certain constraints.

Definition 4.1 $\mathcal{D}(R) = (D_1, D_2, \ldots, D_d)$ is a decomposition of R whose size is d if

(1) $D_1 \cup D_2 \cup \cdots \cup D_d = R$,

and

(2) $D_i \cap D_j = \emptyset$ for any i and j, $i \neq j$.

In order to describe the constraints a decomposition of R must satisfy, a VR-graph is defined which expresses vertical relations between trunks.

Definition 4.2 A VR-graph $G_v = (R, A_v)$ is a directed graph with the vertex set R and the arc set $A_v = \{ (t_i, t_j) \mid L(t_i) < U(t_j) \}$. In other words, an arc from t_i to t_j means that the trunk t_i can be placed above the trunk t_j.

When we want to place two trunks t_i and t_j on the same track, first, we check the VR-graph for the existence of an arc connecting t_i and t_j, and then check the HC-graph for the existence of any path between t_i and t_j. If we find such an arc in the VR-graph and no paths in the HC-graph,

the two trunks may be laid on the same track. If, however, there should be such a path in the HC-graph, then one of these trunks must be placed to the left of the other. The above-mentioned process is intricated, so some essential informations should be transferred from the HC-graph onto arcs of the VR-graph, defining two weights c and w for every arc. In the following, four kinds of decompositions of R are defined by making use of these weights, and a procedure is presented to reduce the size of the optimum decomposition of R at each step.

Example 4.1 Consider a set of nets given in Table 4.1. These nets can be realized as shown in Fig. 4.1, for example. This pattern must be improved, since it contains five tracks but the trunk-crossing count is four. Therefore, some trunk must be divided to reduce the track count.

The HC-graph and VR-graph are shown in Fig. 4.2 and Fig. 4.3, respectively.

t_i	$T_L(t_i)$	$T_R(t_i)$	$U(t_i)$	$L(t_i)$
t_1	{1, 3}	$\emptyset$	1	3
t_2	{9}	{1, 6}	1	9
t_3	$\emptyset$	{9}	9	∞
t_4	{6, 8}	{2}	0	8
t_5	$\emptyset$	{5, 8}	5	8
t_6	{4, 5}	{7}	0	7

Table 4.1. A set of nets $R_{4,1}$.

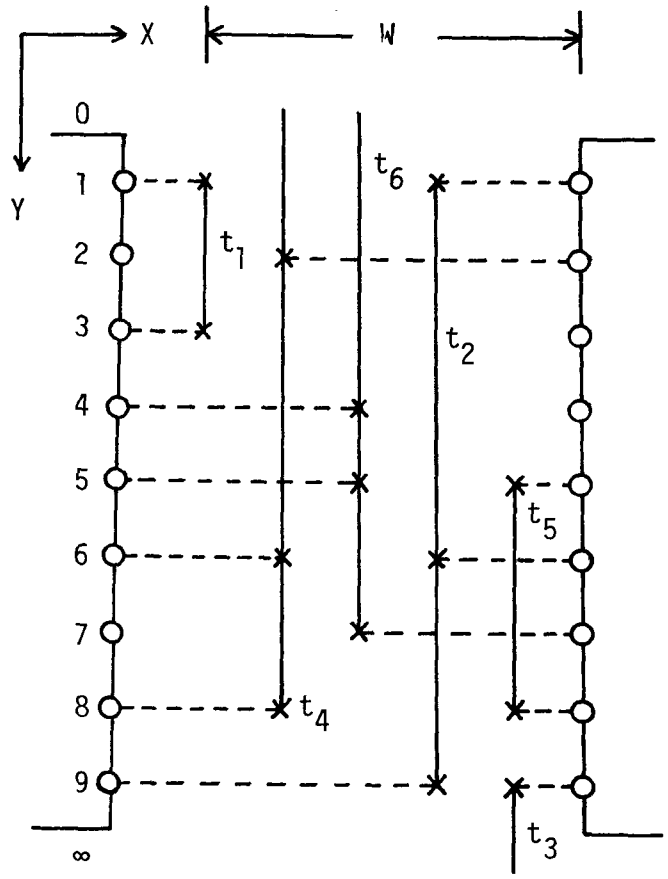


Fig. 4.1. An example of a wiring pattern for $R_{4,1}$.

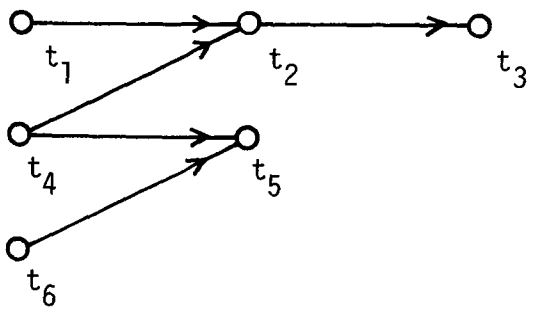


Fig. 4.2. HC-graph for the set of trunks $R_{4,1}$.

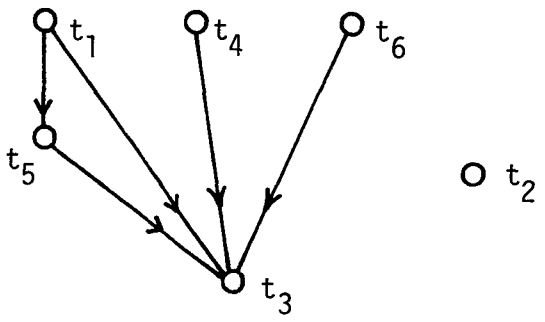


Fig. 4.3. VR-graph for the set of trunks $R_{4,1}$.

4.2.1 a-decomposition of R

Definition 4.3 (a-set) (i) For any trunk t_i, the set $\{ t_i \}$ is an a-set. (ii) For an a-set P and a trunk t_j such that $t_j \notin P$, if $(t_i, t_j) \in A_v$ for any trunk $t_i \in P$, then the set $P \cup \{t_j\}$ is also an a-set.

An a-decomposition of R is a decomposition every element of which is an a-set. The optimum a-decomposition is one of the minimum size (written as $d_a(R)$), and $d_a(R; S)$ denotes the size of the optimum a-decomposition of a subset of R. Here it is evident that $d_a(R) = W_0(R)$, since an a-decomposition is free from horizontal constraints.

For the set of trunks $R_{4,1}$ given in Example 4.1, for example, the optimum a-decompositions are as follows:

$$\mathcal{D}_1(R) = (\{t_1, t_3, t_5\}, \{t_4\}, \{t_6\}, \{t_2\}),$$
$$\mathcal{D}_2(R) = (\{t_1, t_5\}, \{t_4, t_3\}, \{t_6\}, \{t_2\}),$$
$$\mathcal{D}_3(R) = (\{t_1, t_5\}, \{t_4\}, \{t_6, t_3\}, \{t_2\}).$$

4.2.2 c-decomposition of R

As mentioned earlier, two trunks t_i and t_j can be placed on the same track if and only if the VR-graph contains an arc connecting t_i and t_j and the HC-graph does not contain any path from t_i to t_j or one from

t_j to t_i. The above-mentioned informations are represented by defining a weight c for every pair of trunks, as follows:

$$c(t_i, t_j) = \begin{cases} 1 & \text{if } (t_i, t_j) \in A_v \text{ and } \hat{\Gamma}t_i \cup \hat{\Gamma}^{-1}t_i \not\ni t_j, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4.4 (c-set) (i) For any trunk t_i, the set $\{ t_i \}$ is a c-set. (ii) For a c-set P and a trunk t_j such that $t_j \notin P$, if $c(t_i, t_j) = 1$ for any $t_i \in P$, then the set $P \cup \{t_j\}$ is also a c-set.

A c-decomposition of R is a decomposition every element of which is a c-set. The optimum c-decomposition of R and the notation $d_c(R)$ and $d_c(R; S)$ are defined in a similar fashion.

Proposition 4.1 Let $(t_i, t_j) \in A_v$. Then, the trunks t_i and t_j can be placed on the same track if and only if $c(t_i, t_j) = 1$.

(Proof omitted)

Definition 4.5 An arc (t_i, t_j) in a VR-graph is called a c-arc if and only if $c(t_i, t_j) = 0$, and is called a c-free arc if and only if $c(t_i, t_j) = 1$.

Now, reconsider Example 4.1. As is easily seen, the arcs (t_1, t_3) and (t_4, t_3) are c-arcs. So, the decomposition $\mathcal{D}_1(R_{4,1})$ and $\mathcal{D}_2(R_{4,1})$ are not c-decompositions, but $\mathcal{D}_3(R_{4,1})$ is still the optimum c-decomposition. Thus, $d_c(R_{4,1}) = d_a(R_{4,1}) = W_0(R_{4,1})$, while the wiring width of the pattern shown in Fig. 4.1 is greater than $W_0(R_{4,1})$. The following discusses why this extra track is needed.

4.2.3 w-decomposition of R

For every pair of trunks (t_i, t_j), the weight $w(t_i, t_j)$ is defined as follows:

$$w(t_i, t_j) = \begin{cases} 1 & \text{if } c(t_i, t_j) = 1 \text{ and } W(t_i, t_j) \leq W_0(R), \\ 0 & \text{otherwise,} \end{cases}$$

where

$$W(t_i, t_j) = \max(\tilde{x}_L(t_i) + \tilde{x}_R(t_j) + 1, \tilde{x}_R(t_i) + \tilde{x}_L(t_j) + 1),$$

and for every trunk t_k

$$\tilde{x}_L(t_k) = d_c(R; \hat{\Gamma}^{-1}t_k) \text{ and } \tilde{x}_R(t_k) = d_c(R; \hat{\Gamma}t_k).$$

From the definitions above, it is easily recognized that the width required for arranging those trunks which must be placed to the left of the trunk t_k, that is, those belonging to the set $\hat{\Gamma}^{-1}t_k$, can not be less than $\tilde{x}_L(t_k)$, and the width for the set $\hat{\Gamma}t_k$ whose elements must be to the right of the trunk t_k can not be less than $\tilde{x}_R(t_k)$.

Proposition 4.2 Let W be a wiring width and let $x(t_i)$ be an x-coordinate of a track on which the trunk t_i can be placed. Then,

$$\tilde{x}_L(t_i) + 1 \leq x(t_i) \leq W - \tilde{x}_R(t_i).$$

(Proof omitted)

Proposition 4.3 Let $W(t_i, t_j) > W_0(R)$. Then, there are no such wiring patterns with the width $W_0(R)$ as placing t_i and t_j on the same track.

(Proof omitted)

A w-set and a w-decomposition are similarly defined. Here it should be noted that a w-decomposition of R is not always realizable. The statement may be restated as follows: A wire routing process corresponding to a w-decomposition $\mathcal{D}(R)$ is performed in such a way as to place each element of $\mathcal{D}(R)$ on one track. However, this operation may happen to cause a cycle in the order of arranging those tracks.

Again, for Example 4.1, $\tilde{x}_L$ and $\tilde{x}_R$ values are as follows.

The arcs (t_1, t_5) and (t_6, t_3) are w-arcs (an arc (t_i, t_j) of a VR-graph is called a w-arc if $w(t_i, t_j) = 0$) and the decomposition $\mathcal{D}_3(R_{4,1})$ given in 4.2.1 is not a w-decomposition of $R_{4,1}$. Consequently, it is

known that there is no w-decomposition of $R_{4,1}$ whose size is equal to $W_0(R_{4,1})$.

t_i	t_1	t_2	t_3	t_4	t_5	t_6
$\tilde{x}_L(t_i)$	0	2	3	0	2	0
$\tilde{x}_R(t_i)$	2	1	0	2	0	1

Table 4.2. $\tilde{x}_L$ and $\tilde{x}_R$ values.

4.2.4 r-decomposition of R

Realizability of a c-(or w-)decomposition $\mathcal{D}(R) = (D_1, D_2, \ldots, D_d)$ can be checked as follows: First, a decomposed graph $G_D(\mathcal{D}(R), \gamma)$ is constructed, where a directed arc is drawn from a vertex D_i to D_j if and only if $\Gamma D_i \cap D_j \neq \emptyset$, that is,

$\gamma D_i \ni D_j$ if and only if $\Gamma D_i \cap D_j \neq \emptyset$.

Next, the decomposed graph is examined for the existence of any cycle. Then, the necessary and sufficient condition for the decomposition $\mathcal{D}(R)$ to be realizable is that the corresponding decomposed graph does not contain any cycle. And a realizable c-(or w-)decomposition of R is called an r-decomposition. An example of a decomposed graph is shown in Fig. 4.4, which indicates that the c-decomposition $\mathcal{D}_3(R_{4,1}) = (\{t_1, t_5\}, \{t_4\}, \{t_6, t_3\}, \{t_2\})$ is not realizable.

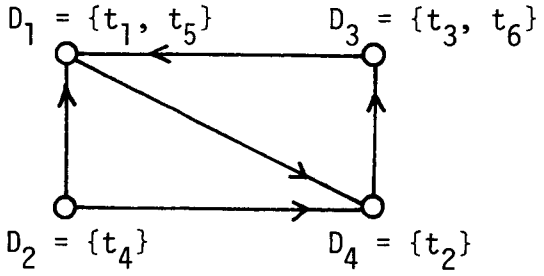


Fig. 4.4. Decomposed graph $G_D = (\mathcal{D}_3(R_{4,1}), \gamma)$.

4.3 Procedures for Reducing Width

From the definitions in the previous chapter, it follows that in general

$$d_r(R) \geqq d_w(R) \geqq d_c(R) \geqq d_a(R) = W_0(R).$$

Hence, the following cases are to be considered:

(1) $d_r(R) = d_w(R) = d_c(R) = d_a(R) = W_0(R)$,

(2) $d_r(R) \geqq d_w(R) \geqq d_c(R) > d_a(R) = W_0(R)$,

(3) $d_r(R) \geqq d_w(R) > d_c(R) = d_a(R) = W_0(R)$,

(4) $d_r(R) > d_w(R) = d_c(R) = d_a(R) = W_0(R)$.

The following discussion deals with the cases of (2), (3), and (4), since the case (1) needs no more improvement.

4.3.1 Procedure for the case $d_c(R) > d_a(R)$

In this case it will be required to reduce the d_c value by dividing some trunks. It seems, however, to be difficult to obtain the sufficient condition under which dividing a trunk causes the d_c value to decrease, in other words, to determine global effects of dividing a trunk. So this thesis takes an approach toward this problem from a standpoint of

evaluating local effects of dividing a trunk.

Consider a subset S of $\mathcal{R}$ such that $d_c(\mathcal{R}; S) > d_a(\mathcal{R}; S)$ but that a division of a certain trunk may lead to the result $d_c(\mathcal{R}'; S') = d_a(\mathcal{R}'; S')$. Since for any subset S the value $d_c(\mathcal{R}'; S')$ is always greater than or equal to 2, a minimal subset having the above-mentioned property is a c- incomplete subset S as defined by

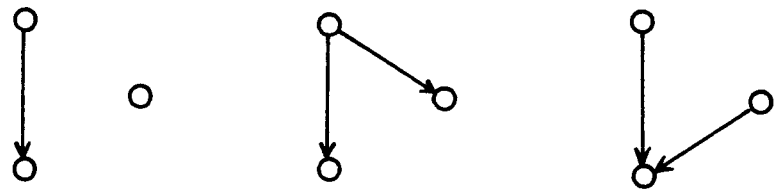
$\#S = 3$ and $d_c(\mathcal{R}; S) = 3 > d_a(\mathcal{R}; S) = 2$.

Theorem 4.1 If $d_c(\mathcal{R}) > d_a(\mathcal{R})$, then there exists a c-incomplete subset S of $\mathcal{R}$.

(Proof) Since $d_c(\mathcal{R}) > d_a(\mathcal{R})$, there exists a c-arc (t_i, t_j), i.e., $c(t_i, t_j) = 0$. From the definition of c, $t_j \in \hat{\Gamma} t_i$ or $t_i \in \hat{\Gamma} t_j$. Here $t_j \in \hat{\Gamma} t_i$ may be assumed without loss of generality. Since $(t_i, t_j) \in A_v$, that is, $U(t_i) < L(t_i) < U(t_j) < L(t_j)$, t_j can not belong to the set Γt_i. It follows that there exists a trunk $t_k \in \Gamma t_i \cap \hat{\Gamma}^{-1} t_j$ and therefore $(t_i, t_k) \notin A_v$, $(t_k, t_i) \notin A_v$ and $c(t_j, t_k) = c(t_k, t_j) = 0$. Consequently, the set $S = \{t_i, t_j, t_k\}$ is c-incomplete.

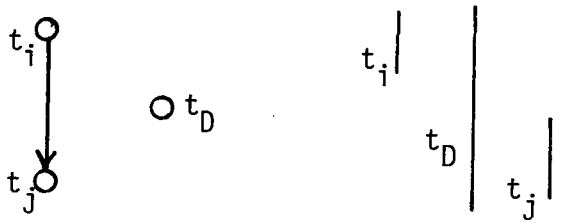
Q. E. D.

The converse of the above theorem is not valid in general.



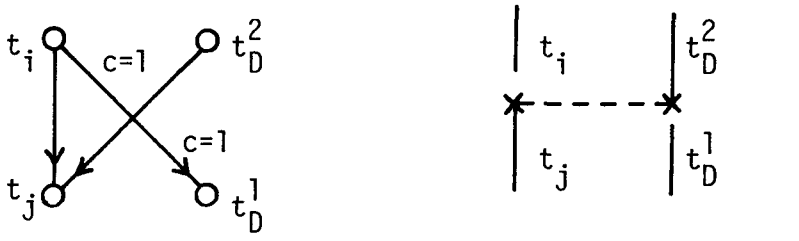
(a) (b) (c)

Fig. 4.5. c-incomplete subsets.



(a) Subgraph (b) Position of trunks

Fig. 4.6. Situation considered in Theorem 4.2.



(a) Subgraph (b) Position of trunks

Fig. 4.7. Effect of dividing the trunk t_D.

Fig. 4.5 shows subgraphs of a VR-graph concerning c-incomplete subsets, forms of which are restricted to those three kinds. For each case (a), (b), and (c) in Fig. 4.5, the conditions for a division of a trunk in a c-incomplete subset to cause $d_c(R'; S') = 2$ may take different forms, according to which trunk in S should be divided. The following considers the situation deemed to be the most effective, as shown in Fig. 4.6, where t_D represents the trunk to be divided. Since the

arrangement of the trunks in S is such one as shown in Fig. 4.6-(b), it appears to be the most desirable to get the situation as shown in Fig. 4.7 by dividing the trunk t_D by the method I or I'. Then it must be noted that either $t_j \in \hat{\Gamma} t_i$ or $t_i \in \hat{\Gamma} t_j$ holds, since $(t_i, t_j) \in A_v$ and $c(t_i, t_j) = 0$. If $t_j \in \hat{\Gamma} t_i$ then t_D must be divided by the method I, and if $t_i \in \hat{\Gamma} t_j$ then t_D must be divided by the method I'.

Theorem 4.2 Assume the situation as shown in Fig. 4.6 and that $t_j \in \hat{\Gamma} t_i$. Then, dividing the trunk t_D at $y = y_D$ by the method I causes the relation $c(t_i, t_D^1) = c(t_D^2, t_j) = 1$ if the following conditions from (1) to (8) are satisfied:

(1) $y_D \in \xi[f_L(t_i, t_D), f_U(t_D, t_j)] \cap \xi[L(t_i), U(t_j)]$,(*)

(2) $t_D \notin (\hat{\Gamma} t_i \cap \hat{\Gamma} t_j) \cup (\hat{\Gamma}^{-1} t_i \cap \hat{\Gamma}^{-1} t_j)$,

(3) $t_L(y_D) \notin \hat{\Gamma} t_D$, (4) $t_R(y_D) \notin \hat{\Gamma}^{-1} t_D$,

(5) $t_L(y_D) \notin \hat{\Gamma} t_i$, (6) $t_R(y_D) \notin \hat{\Gamma}^{-1} t_j$,

(7) $t_R(y_D) \notin \hat{\Gamma}^{-1} t_i$, (8) $t_L(y_D) \notin \hat{\Gamma} t_j$.

Here, if $t_L(y_D)$ or $t_R(y_D)$ is undefined or equal to t_D, then the conditions which include it can be omitted.

(*)

$$\xi[m, n] = \begin{cases} \{m+1, m+2, \ldots, n-1\} & \text{if } n - m \geq 2, \\ \emptyset & \text{otherwise.} \end{cases}$$

$$f_L(t_i, t_D) = \begin{cases} \max R(P(t_i, t_D) \cap \Gamma^{-1} t_D, t_D) & \text{if } t_D \in \hat{\Gamma} t_i, \\ L(t_i) & \text{otherwise.} \end{cases}$$

$$f_U(t_D, t_j) = \begin{cases} \min R(P(t_D, t_j) \cap \Gamma t_D, t_D) & \text{if } t_j \in \hat{\Gamma} t_D, \\ U(t_j) & \text{otherwise.} \end{cases}$$

$$P(t_i, t_j) = (\hat{\Gamma} t_i \cup \{t_i\}) \cap (\hat{\Gamma}^{-1} t_j \cup \{t_j\})$$

(Proof) First, the necessity of the conditions from (2) to (8) is verified.

if not (2), at least one of t_D^1 and t_D^2 is not in the set $(\hat{\Gamma}t_i \cap \hat{\Gamma}t_j) \cup (\hat{\Gamma}^{-1}t_i \cap \hat{\Gamma}^{-1}t_j)$ wherever t_D is divided. Hence, either $c(t_i, t_D^1) = 0$ or $c(t_D^2, t_j) = 0$ holds.

If not (3), for a similar reason, either $t_L(y_D) \in \hat{\Gamma}t_D^1$ or $t_R(y_D) \in t_D^2$ is valid. Hence, a cycle arises in the HC-graph resulting from the division of t_D. The case of (4) is similar to this.

In general, $\hat{\Gamma}t_L(y_D) \supseteq \{t_D^1, t_D^2\}$ and $\hat{\Gamma}^{-1}t_R(y_D) \supseteq \{t_D^1, t_D^2\}$. It follows that if not (5) or if not (7) then $c(t_i, t_D^1) = 0$ holds, and that if not (6) or if not (8) then $c(t_D^2, t_j) = 0$ holds.

Next to consider is the condition (1). It may be assumed that $y_D \in \xi[L(t_i), U(t_j)]$, since the dividing point y_D must be chosen between $L(t_i) + 1$ and $U(t_j) - 1$. Now, we have only to deal with the following case:

$$\xi[f_L(t_i, t_D), f_U(t_D, t_j)] = \emptyset,$$

that is,

$$f_U(t_D, t_j) - f_L(t_i, t_D) \leq 1. \tag{4-1}$$

Then, since $U(t_j) - L(t_i) \geq 2$ from the assumption, either $f_U(t_D, t_j) \neq U(t_j)$ or $f_L(t_i, t_D) \neq L(t_i)$ must hold, that is, $t_D \in \hat{\Gamma}t_i$ or $t_j \in \hat{\Gamma}t_D$ must hold.

(a) The case of $t_D \in \hat{\Gamma}t_i$ and $t_j \notin \hat{\Gamma}t_D$

Evidently,

$$f_L(t_i, t_D) = \max R(P(t_i, t_D) \cap \Gamma^{-1}t_D, t_D) \tag{4-2}$$

and

$$f_U(t_D, t_j) = U(t_j). \tag{4-3}$$

Combining the above equations yields

$U(t_j) - f_L(t_i, t_D) \leq 1$.

Hence, $y_D \leq f_L(t_i, t_D)$ holds since $y_D < U(t_j)$. Consider any trunk $t_p \in P(t_i, t_D) \cap \Gamma^{-1}t_D$; from the above relation and the definition of $f_L(t_i, t_D)$, it must be valid that

$\min(T_L(t_p) \cap T_R(t_D)) \geq y_D$.

Consequently, it is seen that there still remains the relation $t_D^1 \in \hat{\Gamma}t_p$ after dividing t_D at $y = y_D$ by the method I. Therefore, $c(t_i, t_D^1) = 0$ holds since $t_p \in \hat{\Gamma}t_i$.

(b) The case of $t_D \notin \hat{\Gamma}t_i$ and $t_j \in \hat{\Gamma}t_D$

This is a dual situation of the above, and $c(t_D^2, t_j) = 0$ is obtained.

(c) The case $t_D \in \hat{\Gamma}t_i$ and $t_j \in \hat{\Gamma}t_D$

The proof is obvious from the above (a) and (b).

Q. E. D.

Theorem 4.3 Assume the situation as shown in Fig. 4.6 and that $t_j \in \hat{\Gamma}t_i$. Then, dividing the trunk t_D at $y = y_D$ by the method I' causes the relation $c(t_i, t_D^1) = c(t_D^2, t_j) = 1$ if the following conditions from (1) to (8) are satisfied:

(1) $y_D \in \xi[f'_L(t_i, t_D), f'_U(t_D, t_j)]$,(*)

(2) $t_D \notin (\hat{\Gamma}t_i \cap \hat{\Gamma}t_j) \cup (\hat{\Gamma}^{-1}t_i \cap \hat{\Gamma}^{-1}t_j)$,

(3) $t_L(y_D) \notin \hat{\Gamma}t_D$, (4) $t_R(y_D) \notin \hat{\Gamma}^{-1}t_D$,

(5) $t_L(y_D) \notin \hat{\Gamma}t_j$, (6) $t_R(y_D) \notin \hat{\Gamma}^{-1}t_i$,

---

(*)

$$f'_L(t_i, t_D) = \begin{cases} \max R(P(t_D, t_i) \cap \Gamma t_D, t_D) & \text{if } t_i \in \hat{\Gamma}t_D, \\ L(t_i) & \text{otherwise.} \end{cases}$$

$$f'_U(t_D, t_j) = \begin{cases} \min R(P(t_j, t_D) \cap \Gamma^{-1}t_D, t_D) & \text{if } t_D \in \hat{\Gamma}t_j, \\ U(t_j) & \text{otherwise.} \end{cases}$$

(7) $t_R(y_D) \notin \hat{\Gamma}^{-1}t_j$, (8) $t_L(y_D) \notin \hat{\Gamma}t_i$.

Here, $t_L(y_D)$ or $t_R(y_D)$ is undefined or equal to t_D, then the conditions which include it can be omitted.

(Proof omitted)

Fig. 4.8 illustrates trunk-division effects for various forms of c-incomplete subsets.

4.3.2 Procedure for the case $d_w(R) > d_a(R)$

In this case a similar procedure can be considered. A w-incomplete subset S of R is defined by

$\#S = 3$ and $d_w(R; S) = 3 > d_a(R; S) = 2$.

Here it should be noted that a theorem similar to Theorem 4.1 is not valid for the case $d_w(R) > d_a(R)$. In fact, there is an example such that $d_w(R) > d_a(R)$ but there is no w-incomplete subset. Such a set of trunks, however, has a strictly restricted VR-graph.

Theorem 4.4 Let $d_w(R) > d_a(R)$. If there are no w-incomplete subsets in the VR-graph corresponding to a set of nets R, then the following statements are valid:

(1) For every arc (t_i, t_j) in the VR-graph, $c(t_i, t_j) = 1$.

(2) For every trunk t_i, $\tilde{x}_L(t_i) < W_0(R)$ and $\tilde{x}_R(t_i) < W_0(R)$.

(3) If $\tilde{x}_L(t_i) + \tilde{x}_R(t_j) + 1 > W_0(R)$ for an arc (t_i, t_j), then, $\Gamma t_i = \Gamma^{-1}t_j = \emptyset$ and $w(t_p, t_j) = w(t_i, t_q) = 1$ for any $t_p \in \hat{\Gamma}^{-1}t_i$ and $t_q \in \hat{\Gamma}t_j$.

(4) If $\tilde{x}_R(t_i) + \tilde{x}_L(t_j) + 1 > W_0(R)$ for an arc (t_i, t_j), then $\Gamma^{-1}t_i = \Gamma t_j = \emptyset$ and $w(t_p, t_j) = w(t_i, t_q) = 1$ for any $t_p \in \hat{\Gamma}t_i$ and $t_q \in \hat{\Gamma}^{-1}t_j$.

(Proof) The proof proceeds by contradiction.

(1) If there exists an arc (t_i, t_j) such that $c(t_i, t_j) = 0$, there is a c-incomplete subset by Theorem 4.1. Hence, we can conclude that

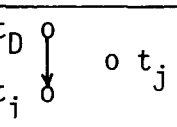
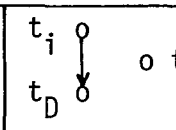
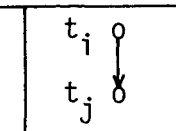
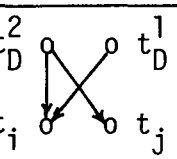
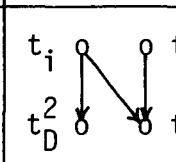
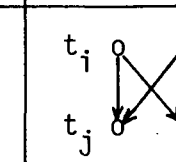
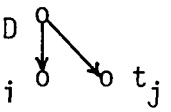
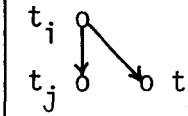
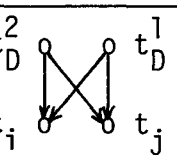
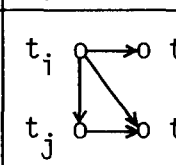
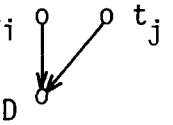
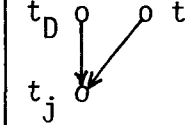
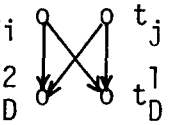
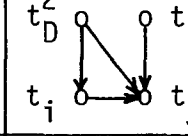
(1)			
(2)			
(3)	$c(t_D^2,t_j)=1$ $c(t_D^1,t_i)=1$	$c(t_i,t_D^2)=1$ $c(t_j,t_D^1)=1$	$c(t_i,t_D^1)=1$ $c(t_D^2,t_j)=1$
	(a-1)	(a-2)	(a-3)
(1)			
(2)			
(3)	$c(t_D^2,t_j)=1$ $c(t_D^1,t_i)=1$	$c(t_i,t_D^2)=1$ $c(t_j,t_D^1)=1$	
	(b-1)	(b-2)	
(1)			
(2)			
(3)	$c(t_i,t_D^1)=1$ $c(t_j,t_D^2)=1$	$c(t_D^2,t_i)=1$ $c(t_D^1,t_j)=1$	
	(c-1)	(c-2)	

Fig. 4.8. Trunk-division effects for various forms of c-incomplete subsets, (1) before division, (2) after division and (3) result.

there is a w-incomplete subset, since any c-incomplete subset is also w-incomplete.

(2) Assume the condition (1). Then, it must be true that if $(t_i, t_j) \in A_v$ then $\hat{\Gamma}t_i \cup \hat{\Gamma}^{-1}t_i \not\ni t_j$. Accordingly, if there exists a trunk t_i such that $\tilde{x}_L(t_i) \geq W_0(R)$, we have

$$d_c(R; \hat{\Gamma}^{-1}t_i \cup \{t_i\}) = d_c(R; \hat{\Gamma}^{-1}t_i) + 1 = \tilde{x}_L(t_i) + 1$$

$$> W_0(R) = d_c(R),$$

which is a contradiction. Similar matters are true in case of $\tilde{x}_R(t_i) > W_0(R)$.

(3) Assume the conditions (1) and (2) and that there is an arc (t_i, t_j) such that

$$\tilde{x}_L(t_i) + \tilde{x}_R(t_j) + 1 > W_0(R) \text{ and } \Gamma t_i \neq \emptyset.$$

Then, for any trunk t_k in Γt_i,

$$(t_i, t_k) \notin A_v \text{ and } (t_k, t_i) \notin A_v$$

and since $\tilde{x}_L(t_k) > \tilde{x}_L(t_i)$,

$$w(t_k, t_j) = w(t_j, t_k) = 0.$$

Consequently, the subset $\{t_i, t_j, t_k\}$ is w-incomplete.

Next to consider is the latter half of the condition (3). Consider arbitrary trunks t_p and t_q such that $t_p \in \hat{\Gamma}^{-1}t_i$ and $t_q \in \hat{\Gamma}t_j$: It follows from the condition (1) that t_p is detached from t_i and that t_q is detached from t_j in the VR-graph. We can easily see that $(t_j, t_p) \notin A_v$ and $(t_q, t_i) \notin A_v$. Suppose $(t_j, t_p) \in A_v$, and the two arcs (t_i, t_j) and (t_j, t_p) of the VR-graph guarantees the existence of an arc (t_i, t_p). This contradicts the condition (1) mentioned above. Similar matters are true for the pair of t_q and t_i. Thus, if $w(t_p, t_j) = 0$ or $w(t_i, t_q) = 0$ holds, then $\{t_i, t_j, t_p\}$ or $\{t_i, t_j, t_q\}$ is w-incomplete, respectively.

The remaining part of the proof is similar to the above.

Q. E. D.

Example 4.2 A set of nets $R_{4,2}$, given in Table 4.3, has the properties stated in the above theorem.

HC-graph and VR-graph are shown in Fig. 4.9 and Fig. 4.10, respectively. For clarity, transitive arcs are omitted in the VR-graph.

t_i	$T_L(t_i)$	$T_R(t_i)$	$U(t_i)$	$L(t_i)$
t_1	{12}	∅	0	12
t_2	{13}	∅	13	∞
t_3	{2, 11}	∅	2	11
t_4	∅	{12}	12	∞
t_5	{3, 7}	∅	3	7
t_6	{1}	{2, 3}	1	3
t_7	∅	{1, 5}	1	5
t_8	{9}	{4}	4	9
t_9	∅	{6, 14}	6	14
t_{10}	{15, 17}	∅	15	∞
t_{11}	{10}	{17}	10	17
t_{12}	∅	{8, 10}	8	∞

Table 4.3. A set of nets $R_{4,2}$.

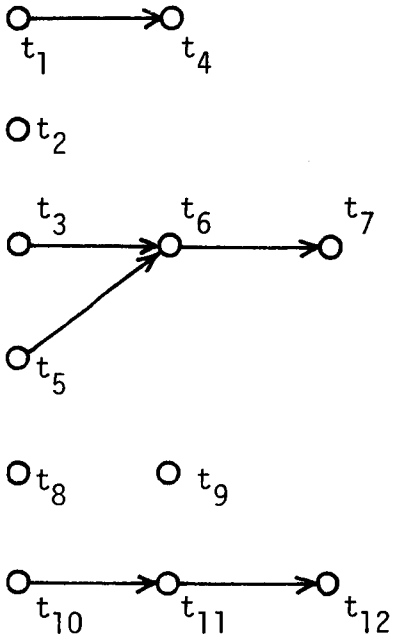


Fig. 4.9. The HC-graph $G_h(R_{4,2}, \Gamma)$.

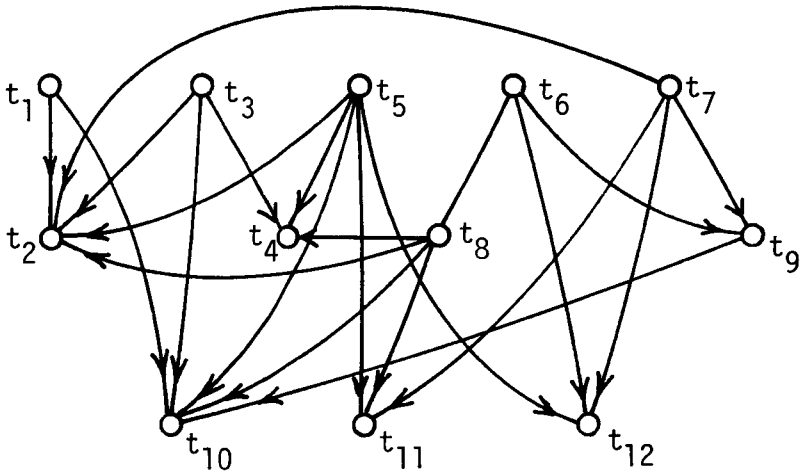


Fig. 4.10. The VR-graph $G_v(R_{4,2}, A_v)$.

For the set of nets $R_{4,2}$, $W_0(R_{4,2}) = 5$ and (t_7, t_{10}) is the only w-arc of the VR-graph (of course, there are no c-arcs). Also, we can see that a decomposition

$$\mathcal{D}(R_{4,2}) = (\{t_1, t_2\}, \{t_3, t_4\}, \{t_5, t_{12}\}, \{t_6, t_8, t_{11}\}, \{t_7, t_9, t_{10}\})$$

is the only one optimum c-decomposition of $R_{4,2}$, and $d_c(R_{4,2}) = d_a(R_{4,2}) = W_0(R_{4,2}) = 5$ and $d_r(R_{4,2}) = d_w(R_{4,2}) = 6$.

Procedures for w-incomplete subsets are similar for the most part to the ones for c-incomplete subsets and will be implemented while any w-incomplete subset remains to be processed.

4.3.3 Procedure for the case $d_r(R) > d_w(R)$

The relation $d_r(R) > d_w(R)$ means that when wire routing is performed accordingly to an optimum w-decomposition of R a cycle arises in the order of arranging tracks. Hence, a procedure is necessary which eliminates such a cycle without increasing the value of $d_w(R)$, but the author could not present such a procedure.

4.4 Outline of Width Reduction Process

From the results obtained thus far, the following algorithm may be considered.

[Outline of width reduction process]

begin

START:

if $W(R) = W_0(R)$ then stop;

D_p := a set of possible dividing patterns;

for every dividing pattern $(t_i, y_D, M) \in D_p$ do

if this division does not produce any cycles in the HC-graph

```
then begin
  Σ := a set of incomplete subsets;
  for every incomplete subset S do
    if W(S(t_i, y_D, M)) < W(S) and
    W(R(t_i, y_D, M)) < W(R) then
    begin
      divide the trunk t_i at y = y_D by the method M;
      R := R(t_i, y_D, M);
      go to START
    end
  end
end;
```

In the above algorithm, $S(t_i, y_D, M)$ and $R(t_i, y_D, M)$ represent those sets of trunks which result from the division of the trunk t_i at $y = y_D$ by the method M.

This process is, as is obvious, too inefficient and too tedious, for we may find, in a practical problem, too many dividing patterns to examine their effects, and also too many incomplete subsets. The following section tries to put restrictions on dividing patterns and incomplete subsets in order to improve efficiency.

4.5 Matching Condition

This section considers what restrictions are to be put on dividing patterns and incomplete subsets in the algorithm proposed in the previous section. For this purpose trunk groups are redefined more formally and their interrelations are investigated.

4.5.1 Property of trunk groups

Trunk groups are redefined in a different manner in order to investigate their interrelations in more detail.

It is easily known from the definition of trunk groups given earlier that for two trunks t_i and t_j there exists no trunk group which contains both of them if and only if t_i and t_j do not have any common part in the direction of the y-axis, in other words, if and only if the VR-graph contains an arc (t_i, t_j) or (t_j, t_i).

Proposition 4.4 *A trunk group G_i is a maximal subset of R which has the following property:*

$$\max\{ U(t_j) \mid t_j \in G_i \} \leq \min\{ L(t_j) \mid t_j \in G_i \}.$$

(Proof omitted)

The above property suggests an effective method of determining trunk groups by use of a third directed graph. This method should be distinguished for its capability to express, in a brief form, essential information which is necessary to examine the possibility of reducing wiring width.

Definition 4.6 (Simplified VR-graph)

A simplified VR-graph $G_s = (V_1, V_2, A_s)$ is defined as follows:

(1) The vertex set of G_s is $V_1 \cup V_2$, where V_1 and V_2 are both equal to a set of trunks R, but elements of V_1 are ordered by their lower ends and those of V_2 are ordered by their upper ends. Hereafter, V_1 and V_2 are denoted as

$$V_1 = \{t_1^{(1)}, t_2^{(1)}, \ldots, t_N^{(1)}\},$$

and

$$V_2 = \{t_1^{(2)}, t_2^{(2)}, \ldots, t_N^{(2)}\},$$

where

$$L(t_1^{(1)}) \leqq L(t_2^{(1)}) \leqq \cdots \leqq L(t_N^{(1)}),$$

and

$$U(t_1^{(2)}) \leqq U(t_2^{(2)}) \leqq \cdots \leqq U(t_N^{(2)}).$$

(2) A_s denotes the (directed-) arc set of G_s, and $(t_i^{(1)}, t_j^{(2)}) \in A_s$ if and only if

(i) $L(t_i^{(1)}) < U(t_j^{(2)})$,

(ii) $L(t_k^{(1)}) \geqq U(t_j^{(2)})$ for any k, $k > i$, and

(iii) $L(t_i^{(1)}) \geqq U(t_k^{(2)})$ for any k, $k < j$.

Hereafter, A_s is denoted as

$$A_s = \{(t_{i_k}^{(1)}, t_{j_k}^{(2)}) \mid 1 \leqq k \leqq n\}$$

and sets of trunks $V_1^1, V_2^1, \ldots, V_{n+1}^1$ and $V_0^2, V_1^2, \ldots, V_n^2$ are defined as follows:

$$V_k^1 = \{t_{i_{k-1}+1}^{(1)}, t_{i_{k-1}+2}^{(1)}, \ldots, t_{i_k}^{(1)}\} \quad \text{for any k, } 1 \leqq k \leqq n+1,$$

and

$$V_k^2 = \{t_{j_k}^{(2)}, t_{j_k+1}^{(2)}, \ldots, t_{j_{k+1}-1}^{(2)}\} \quad \text{for any k, } 0 \leqq k \leqq n,$$

where $i_0 = 0$, $i_{n+1} = N$, $j_0 = 1$ and $j_{n+1} = N + 1$.

Then the following sets of trunks $G_1, G_2, \ldots, G_{n+1}$ are trunk groups:

$$G_1 = V_0^2,$$

$$G_2 = (V_0^2 \cup V_1^2) - V_1^1,$$

.....

$$G_k = (\bigcup_{i=0}^{k-1} V_1^2) - (\bigcup_{i=1}^{k-1} V_i^1),$$

.....

$$G_{n+1} = R - (\bigcup_{i=1}^{n} V_i^1).$$

From the above relations, the cardinality of each trunk group is caluculated as follows:

$$\#G_1 = j_1 - 1,$$

$$\#G_2 = j_2 - i_1 - 1,$$

.....

$$\#G_k = j_k - i_{k-1} - 1,$$

.....

$$\#G_{n+1} = N - i_n.$$

4.5.2 Matching condition

The following examines interrelations between trunk groups, in particular, between a max group G_k and its neighboring trunk groups G_{k-1} and G_{k+1}.

Lemma 4.1 $V_a^2 \cap V_b^1 = \emptyset$ for any a and b where $a \geq b$.

(Proof) It follows from the definition of the simplified VR-graph that if $a \geq b$ then

$$U(t_{j_a}^{(2)}) > L(t_{i_a}^{(1)}) > L(t_{i_b}^{(1)}),$$

$$U(t_{j_a}^{(2)}) = \min\{ U(t_k) \mid t_k \in V_a^2 \},$$

$$U(t_{i_b}^{(1)}) = \max\{ L(t_k) \mid t_k \in V_b^1 \}.$$

Consequently, it follows from the above that

$$\min\{ U(t_k) \mid t_k \in V_a^2 \} > \max\{ L(t_k) \mid t_k \in V_b^1 \}.$$

This terminates the proof of this lemma.

Q. E. D.

Lemma 4.2 $G_{k+1} = (G_k - V_k^1) \cup V_k^2$ for any k where $1 \leq k \leq n$.

(Proof) From the equation defining trunk groups,

$$G_k = (\bigcup_{i=0}^{k-1} V_i^2) - (\bigcup_{i=1}^{k-1} V_i^1).$$

and

$$G_{k+1} = (\bigcup_{i=0}^{k} V_i^2) - (\bigcup_{i=1}^{k} V_i^1)$$
$$= [V_k^2 \cup (\bigcup_{i=0}^{k-1} V_i^2) - (\bigcup_{i=1}^{k-1} V_i^1)] - V_k^1.$$

Here, since by Lemma 4.1

$$V_k^2 \cap (\bigcup_{i=0}^{k-1} V_i^1) = \emptyset,$$

the above representation of G_{k+1} is replaced as

$$G_{k+1} = V_k^2 \cup [(\bigcup_{i=0}^{k-1} V_i^2) - (\bigcup_{i=1}^{k-1} V_i^1)] - V_k^1$$
$$= (V_k^2 \cup G_k) - V_k^1.$$

Further, using the relation $V_k^2 \cap V_k^1 = \emptyset$ derived from Lemma 4.1, we have

$$G_{k+1} = (G_k - V_k^1) \cup V_k^2.$$

Q. E. D.

Lemma 4.2' $G_{k-1} = (G_k - V_{k-1}^2) \cup V_{k-1}^1$ for any k where $2 \leq k \leq n+1$.

(Proof) It follows from the definition of G_k that

$$\bigcup_{i=0}^{k-1} V_i^2 = R - (\bigcup_{i=k}^{n} V_i^2),$$

and

$$\bigcup_{i=1}^{k-1} V_i^1 = R - (\bigcup_{i=k}^{n+1} V_i^1).$$

Applying the above relations to

$$G_k = (\bigcup_{i=0}^{k-1} V_i^2) - (\bigcup_{i=1}^{k-1} V_i^1),$$

we have

$$G_k = [R - (\bigcup_{i=k}^{n} V_i^2)] - [R - (\bigcup_{i=k}^{n+1} V_i^1)]$$
$$= (\bigcup_{i=k}^{n+1} V_i^1) - (\bigcup_{i=k}^{n} V_i^2).$$

The remaining part of the proof proceeds in a similar manner to Lemma 4.2.

Q. E. D.

Theorem 4.5 For any k where $2 \leq k \leq n$,

$$d_c(\mathcal{R};\ G_k \cup G_{k+1}) = \max(\#G_k,\ \#G_{k+1}) + d_c(\mathcal{R};\ V_k^1 \cup V_k^2) - \max(\#V_k^1,\ \#V_k^2)$$

and

$$d_c(\mathcal{R};\ G_k \cup G_{k-1}) = \max(\#G_k,\ \#G_{k-1}) + d_c(\mathcal{R};\ V_{k-1}^1 \cup V_{k-1}^2) - \max(\#V_{k-1}^1,\ \#V_{k-1}^2).$$

(Proof) From Lemma 4.2 and the relation $G_k \cap V_k^2 = \emptyset$, we obtain the following three representations:

$$\begin{aligned} G_k \cap G_{k+1} &= G_k \cap [(G_k - V_k^1) \cup V_k^2] \\ &= G_k \cap (G_k - V_k^1) = G_k - V_k^1, \\ G_k - G_{k+1} &= G_k - [(G_k - V_k^1) \cup V_k^2] \\ &= G_k - (G_k - V_k^1) = V_k^1, \\ G_{k+1} - G_k &= [(G_k - V_k^1) \cup V_k^2] - G_k \\ &= [(G_k - V_k^1) - G_k] \cup V_k^2 = V_k^2. \end{aligned}$$

Combining the above three representations yields

$$G_k \cup G_{k+1} = (G_k - V_k^1) \dot{\cup} V_k^1 \dot{\cup} V_k^2,$$

where the symbol $\dot{\cup}$ represents union of mutually disjoint sets.

Similarly, from Lemma 4.2' and the relation $G_k \cap V_{k-1}^1 = \emptyset$, we have

$$G_k \cup G_{k+1} = (G_k - V_{k+1}^2) \dot{\cup} V_{k+1}^2 \dot{\cup} V_{k+1}^1.$$

Setting k = k - 1 in the above representation yields

$$G_k \cup G_{k+1} = (G_{k+1} - V_k^2) \dot{\cup} V_k^2 \dot{\cup} V_k^1.$$

Now, consider two arbitrary trunks t_i and t_j in the set $G_k \cup G_{k+1}$ such that $(t_i, t_j) \in A_v$. Then, it is easily seen that t_i must belong to the set V_k^1 and t_j to V_k^2. Hence, the above two representations imply that

$$\begin{aligned} d_c(R; G_k \cup G_{k+1}) &= d_c(R; G_k - V_k^1) + d_c(R; V_k^1 \cup V_k^2) \\ &= d_c(R; G_{k+1} - V_k^2) + d_c(R; V_k^1 \cup V_k^2). \end{aligned}$$

Here, since $G_k \supseteqq V_k^1$ and $G_{k+1} \supseteqq V_k^2$, we have

$$\begin{aligned} d_c(R; G_k - V_k^1) &= d_c(R; G_k) - d_c(R; V_k^1) \\ &= \#G_k - \#V_k^1 \end{aligned}$$

and

$$d_c(R; G_k - V_k^2) = \#G_k - \#V_k^2.$$

In the above, it is easily seen that if $\#G_k \geqq \#G_{k+1}$ then $\#V_k^1 \geqq \#V_k^2$, and if $\#G_k \leqq \#G_{k+1}$ then $\#V_k^1 \leqq \#V_k^2$ since

$$G_k = (G_k \cap G_{k+1}) \dot{\cup} V_k^1$$

and

$$G_{k+1} = (G_k \cap G_{k+1}) \dot{\cup} V_k^2.$$

As a result, we obtain

$$\begin{aligned} d_c(R; G_k \cup G_{k+1}) &= \max(\#G_k, \#G_{k+1}) \\ &\quad + d_c(R; V_k^1 \cup V_k^2) - \max(\#V_k^1, \#V_k^2). \end{aligned}$$

The other representation of this theorem can be obtained by setting $k = k - 1$ in the above.

Q. E. D.

This theorem suggests that it should be examined at the very first whether, for any max group G_k, the following equations are valid:

$$d_c(R;\ V_k^1 \cup V_k^2) = \max(\#V_k^1,\ \#V_k^2)$$

and

$$d_c(R;\ V_{k-1}^1 \cup V_{k-1}^2) = \max(\#V_{k-1}^1,\ \#V_{k-1}^2) \qquad (k \quad 2).$$

The above condition is called a "matching condition" for the weight c. Similarly, the matching condition for the weight w is stated as follows:

[Matching condition]

For any max group G_k,

$$d_w(R;\ V_k^1 \cup V_k^2) = \max(\#V_k^1,\ \#V_k^2)$$

and

$$d_w(R;\ V_{k-1}^1 \cup V_{k-1}^2) = \max(\#V_{k-1}^1,\ \#V_{k-1}^2).$$

This matching condition is used to limit a set of those trunks which are to be divided. This is done in the following manner: First, for every max group, the matching condition is examined. Then, if $d_w(R;\ V_k^1 \cup V_k^2) > \max(\#V_k^1,\ \#V_k^2)$, a trunk is needed to be divided so that the condition is satisfied. If such a division is successful then its overall effect is examined, in other words, it is seen whether the track count is reduced.

[The second version of width reduction process]

begin

START:

if $W(R) = W_0(R)$ then stop;

construct a simplified VR-graph $G_S(V_1, V_2, A_S)$;

```
ARC := ∅;
for k := 1 step 1 until n do
  if #G_k + d_w(R; V_k^1 ∪ V_k^2) - max(#V_k^1, #V_k^2) > W_0(R) then
      ARC := ARC ∪ { (t_i, t_j) | t_i ε V_k^1, t_j ε V_k^2 and w(t_i, t_j) = 0};
for every (t_i, t_j) ε ARC do
  begin T_D := { t_D | {t_D, t_i, t_j} = S is w-incomplete };
    D_p := a set of all possible dividing patterns of t_D;
    for every dividing pattern (t_D, y_D, M) ε D_p do
      if this division does not produce any cycles in the HC-graph
          and W(S(t_D, y_D, M)) = 2 and W(R(t_D, y_D, M)) < W(R)
      then begin
          divide the trunk t_D at y = y_D by the method M;
          R := R(t_D, y_D, M);
          go to START
        end
  end
end;
```

4.6 Algorithm B

--- Width Reduction Algorithm ---

Based upon the results obtained thus far, this section presents an algorithm for achieving a near-minimum width layout for a given set of trunks. Here it should be noted that the algorithm deals with a set of trunks to which the corresponding HC-graph does not contain any cycles, for Algorithm A, which was presented in the preceding chapter, has

eliminated all cycles in the original HC-graph.

In a manual design, it may be rather easy in a small-scale problem to find a dividing pattern of a trunk which reduces wiring width, if an appropriate layout pattern is present. The reason may be considered to be its easiness to evaluate an effect of a division by means of partial amendment of the layout pattern. There arise many difficulties, however, in expressing the above-mentioned process in a procedural form. First to do is to evaluate a division of a trunk in a suitable way. Algorithm A evaluated a dividing pattern from the aspect of eliminating cyclic constraints, so local evaluation is quite enough. On the other hand in the width reduction process, a dividing pattern is effective only if it really reduces wiring width. Thus, evaluation of a dividing pattern requires the following process: First, divide a trunk. Second, lay out the resulting set of trunks in the minimum possible width. Last, check whether the width is reduced. And then, if the division fails in reducing the width, the divided trunks and the HC-graph must be restored to the pre-division condition.

Evidently, this is a quite tedious procedure. Thus, some suitable local evaluation is needed in order to increase the efficiency of the width reduction process so that trunks are less frequently divided. For this purpose the final version of the width reduction process presented below takes a heuristic search method. A heuristic function F_B is determined for every tuple (t_i, t_j, t_D, y_D, M) where (t_i, t_j) is a w-arc, $\{t_i, t_j, t_D\}$ is a w-incomplete subset and the trunk t_D is divided at $y = y_D$ by the method M.

(1) $F_B(t_i, t_j, t_D, y_D, M) = \omega$ (undefined) if at least one of the followings is the case:

(i) It is impossible to divide the trunk t_D at $y = y_D$ by the method M.

(ii) The division generates cycles in the HC-graph.

(iii) The resulting trunk-crossing count $W_0(R(t_D, y_D, M))$ is greater than W_0, where W_0 is a parameter determined in the algorithm.

(iv) $W(R - \{t_D\})$ is equal to $W(R)$. If $W(R - \{t_D\}) = W(R)$ then any dividing pattern of t_D is of no effect, since for any dividing pattern (t_D, y_D, M), $W(R(t_D, y_D, M)) \geq W(R - \{t_D\})$ holds.

(v) The division cuts off no directed paths of the HC-graph.

(vi) There exist both $t_L(y_D)$ and $t_R(y_D)$, and

$$x_L(t_L(y_D)) + x_R(t_R(y_D)) + 2 > W(R).$$

(vii) $W(\{t_D^1, t_D^2, t_i, t_j\})$ is greater than or equal to $W(\{t_D, t_i, t_j\})$.

(2) Otherwise, $F_B(t_i, t_j, t_D, y_D, M) = \Sigma_{k=5}^{9} c_k \cdot f_k(t_D, y_D, M)$, where $c_5, \ldots, c_9$ are positive constants.

$f_5(t_D, y_D, M)$ = (the length of the longest directed path of the original HC-graph) - (the length of the longest directed path of the HC-graph after the division),

$f_6(t_D, y_D, M)$ = (the sum of the length of those directed paths that are cut off by the division),

$f_7(t_D, y_D, M) = \min(x_L(t_D), x_R(t_D))$,

$f_8(t_D, y_D, M) = f_3(t_D, y_D, M)$,

$f_9(t_D, y_D, M) = f_4(t_D, y_D, M)$,

where f_3 and f_4 are defined in the preceding chapter.

[Algorithm B --- width reduction algorithm]

begin

START:

EXIT1: if $W(R) = W_0(R)$ then stop;

construct a simplified VR-graph $G_S(V_1, V_2, A_S)$;

ARC := $\emptyset$;

comment:

This block examines the matching condition for every trunk group. If the matching condition is not the case for a trunk group G_k, then those w-arcs between V_k^1 and V_k^2 are stored into ARC;

for k := 1 step 1 until n do

if $\#G_k + d_w(R; V_k^1 \cup V_k^2) - \max(\#V_k^1, \#V_k^2) > W_0(R)$ then

begin

ARC := ARC $\cup \{(t_i, t_j) \mid t_i \in V_k^1, t_j \in V_k^2$ and $w(t_i, t_j) = 0\}$;

for every w-arc $(t_i, t_j) \in$ ARC do

$F'_B(t_i, t_j) := \#G_k + d_w(R; V_k^1 \cup V_k^2) - \max(\#V_k^1, \#V_k^2) - W_0(R)$

end;

DPAT := $\emptyset$;

comment:

The following block chooses a third trunk t_D to be divided for every w-arc (t_i, t_j) contained in ARC. Here it should be noted that if $c(t_i, t_j) = 0$ then the trunk t_D must be an articulation point of directed paths connecting t_i and t_j, and otherwise t_D must be divided so as to reduce the value $x_L(t_i) + x_R(t_j) - 1$ or $x_R(t_i) + x_L(t_j) - 1$.

for every $(t_i, t_j) \in$ ARC do

begin

if $c(t_i, t_j) = 0$ then

$T(t_i, t_j) := \{(t_i, t_j, t_D) \mid \{t_i, t_j, t_D\}$ is c-incomplete

and $t_D \in P_a(t_i, t_j) \cup P_a(t_j, t_i)\}$

else

$T(t_i, t_j) := \{(t_i, t_j, t_D) \mid \{t_i, t_j, t_D\}$ is w-incomplete

and $(\hat{\Gamma} t_D \cup \hat{\Gamma}^{-1} t_D) \cap \{t_i, t_j\} \neq \emptyset\}$;

for every $(t_i, t_j, t_D) \in T(t_i, t_j)$ do

DPAT := DPAT $\{(t_i, t_j, t_D, y_D, M) \mid (t_D, y_D, M)$ is a dividing

pattern of the trunk $t_D\}$

end;

$W_0 := W_0(R)$;

TEST := $\emptyset$;

LOOP:

compute $F_B(t_i, t_j, t_D, y_D, M)$ for every $(t_i, t_j, t_D, y_D, M) \in$ DPAT;

$D_p := \{(t_i, t_j, t_D, y_D, M) \mid F_B(t_i, t_j, t_D, y_D, M) \neq \omega\}$;

while $D_p \neq \emptyset$ do

begin

choose an element (t_i, t_j, t_D, y_D, M) from D_p to give the maximum

value for $F'_B + F_B$;

if $(t_D, y_D, M) \in$ TEST then

begin

divide the trunk t_D at $y = y_D$ by the method M;

```
        if W(R(t_D, y_D, M)) < W(R) then
          begin R := R(t_D, y_D, M); go to START end
        else begin
          restore the divided trunk to the pre-division condition;
          TEST := TEST ∪ {(t_D, y_D, M)} end
      end;
      comment:
          TEST remembers those dividing patterns that have ever been
          tried in vain. Thus, if (t_D, y_D, M) ε TEST then the dividing
          pattern does not have to be tried;
      D_p := D_p - {(t_i, t_j, t_D, y_D, M)}
    end;
EXIT2: if W_0 > W_0(R) then stop;
EXIT3: if W_0 = W_0(R) and W(R) = W_0(R) + 1 then stop;
    W_0 := W_0 + 1;
    go to LOOP
  end Algorithm B;
```

Algorithm B contains three exits. If the algorithm terminates at EXIT1, then the track count $W(R)$ is equal to the trunk-crossing count $W_0(R)$ and thus no more procedure is needed. On the other hand if it terminates at the other exits then $W(R) > W_0(R)$. Also, in these cases it is easily seen that the track count $W(R)$ can not be reduced by any dividing patterns of trunks which do not increase the trunk-crossing count $W_0(R)$. Hence, if $W(R) = W_0(R) + 1$ then any more search is insignificant, and thus the algorithm terminates at EXIT3. If $W(R) > W_0(R) + 1$

then the algorithm must continue to search for a dividing pattern which reduces the track count $W(R)$. In this case, all of such dividing patterns increase the trunk-crossing count $W_0(R)$. If the algorithm finds no such patterns then it terminates at EXIT2.

4.7 Algorithm C

--- Algorithm for Arranging Trunks ---

Algorithms A and B determine which trunks to be divided and how to divide them, but they do not show the way to arrange those trunks. Algorithm C to be presented in this section achieves the fewest possible tracks, in other words, it arranges trunks in the minimum possible width. Note that this algorithm also estimates the track count $W(R)$ for a given set of trunks R if the HC-graph is acyclic.

[Algorithm C]

```
begin
  while R ≠ ∅ do
  begin
    construct the HC-graph G_h(R, Γ);
    R_L := { t_i | t_i ε R and Γ^-1 t_i = ∅ };
    if there exists a c-set such that S ⊇ R_L then
    begin
      place those trunks belonging to R_L on the leftmost track;
      R := R - R_L
    end
    else begin
      enumerate those c-sets that are subsets of R_L;
      evaluate F_C values for all those c-sets;
```

```
      let S_L be a c-set for which the F_C value is the maximum;
      R_R := { t_j | t_j ε R and Γt_j = ∅ };
      if there exists a c-set S such that S ⊇ R_R then
      begin
        place those trunks belonging to R_R on the rightmost track;
        R := R - R_R
      end
      else begin
        enumerate those c-sets that are subsets of R_R;
        evaluate F_C values for all those c-sets;
        let S_R be a c-set for which the F_C value is the maximum;
        if F_C(S_L) ≥ F_C(S_R) then
        begin
          place those trunks belonging to S_L on the leftmost track;
          R := R - S_L
        end
        else begin
          place those trunks belonging to S_R on the rightmost track;
          R := R - S_R
        end
      end
    end
  end
end Algorithm C;
  comment
      R_L (or R_R) is a set of those trunks which can be placed on the
      leftmost (or rightmost) track.
```

The heuristic function F_C is determined as follows:

$$F_C(S) = c_{10} \cdot f_{10}(S) - c_{11} \cdot f_{11}(S),$$

where S is a c-set and c_{10} and c_{11} are positive constants.

$$f_{10}(S) = K_1 \cdot \max\{ x_L(t_j) + x_R(t_j) - 1 \mid t_j \in S \} + K_2 \cdot \#S,$$

where K_1 and K_2 are positive constants.

$$f_{11}(S) = \#\bar{S} - \#S,$$

where $\bar{S}$ is the maximum c-set such that $\bar{S} \subseteqq R$ and $\bar{S} \supseteqq S$.

The following is the discussion on the heuristic function F_C. For a trunk t_i, the larger is the value $x_L(t_i) + x_R(t_i) - 1$, the more restricted are those tracks on which the trunk t_i may be placed, as is seen in Property 3.3. This is reflected on the function f_{10}. Next, consider the evaluation by the function f_{11}. Now, let S be a c-set such that $S \subseteqq R_L$ and let $\bar{S}$ be the largest c-set such that $\bar{S} \subseteqq R$ and $\bar{S} \supseteqq S$. Then, if S is much smaller than $\bar{S}$ then it is desirable that the present cycle of the algorithm does not choose S, since some future cycle may happen to find a larger c-set which includes S.

4.8 Computational Results

The algorithm described in this thesis has been programmed with some modifications and run on a FACOM 230-45/S computer. This program is written in FORTRAN IV and consists of about 3K steps. The author has experimented with this program for several wire-routing problems. The following examples demonstrates its usefulness and versatility. All the examples have been artificially designed, because the author could not obtain real designs. Real problems, however, seem to be easier than these problems considered here.

Example 1

An HC-graph for Example 1 is shown in Fig. 4.11. As is easily seen, the HC-graph contains five cycles $\{t_1, t_2, t_3, t_4, t_5, t_6\}$, $\{t_1, t_2, t_3, t_6\}$, $\{t_2, t_3, t_4, t_5\}$, $\{t_2, t_3, t_6, t_7, t_8, t_5\}$ and $\{t_5, t_6, t_7, t_8\}$.

Algorithm A first enumerates all of those dividing patterns that eliminate the most cycles, and the results are shown in Table 4.6.

It should be noted that the methods I, I' II, III, and III' are represented simply as 1, 2, 3, 4, and 5 in Table 4.6. For example, the first line means that the dividing pattern (t_2, 1, III) eliminates exactly two cycles. Here also note that the table excludes those dividing patterns that increase the trunk-crossing count, for example, a dividing pattern (t_5, 17, II) which eliminates four cycles.

Next, Algorithm A computes the $f_3 + f_4$ value for every dividing pattern and chooses the dividing pattern (t_2, 3, III), as is shown in Table 4.7.

Fig. 4.12 shows the HC-graph resulting from the division of the trunk t_2 at y = 3 by the method III, where t_2 and t_{13} represent t_2^1 and t_2^2, respectively.

At the next cycle of Algorithm A the dividing pattern (t_5, 10, III) is chosen. In this turn Algorithm A must compute the values for $f_2 + f_3 + f_4$ because those dividing patterns enumerated in Table 4.8 can eliminate the only remaining cycle $\{t_2, t_3, t_4, t_5\}$.

Table 4.8 indicates that the dividing pattern (t_4, 15, III') is the optimum and the trunk t_4 is divided at y = 15 by the method III'. The resulting HC-graph is shown in Fig. 4.13. As is easily seen, the HC-graph is acyclic and hence Algorithm A terminates.

Then the control is transferred from Algorithm A to Algorithm B. First, x_L and x_R values must be computed. Trunk groups are constructed by the simplified VR-graph shown in Fig. 4.14.

Trunk groups are as follows:

$G_1 = \{t_1, t_{13}, t_6, t_7, t_2, t_5\}$,

$G_2 = \{t_1, t_6, t_7, t_2, t_5, t_{10}, t_{11}\}$,

$G_3 = \{t_6, t_7, t_2, t_5, t_{10}, t_{11}, t_3\}$,

$G_4 = \{t_7, t_5, t_{10}, t_{11}, t_3, t_4, t_{15}\}$,

$G_5 = \{t_7, t_{10}, t_{11}, t_3, t_4, t_{15}, t_8\}$,

$G_6 = \{t_{10}, t_{11}, t_3, t_4, t_{15}, t_8, t_{14}\}$,

$G_7 = \{t_{10}, t_{11}, t_4, t_{15}, t_8, t_{14}, t_{12}\}$,

$G_8 = \{t_{11}, t_4, t_{15}, t_8, t_{14}, t_{12}, t_9\}$.

All except the trunk group G_1 are max groups. The values for x_L and x_R are computed by using the above trunk groups.

Algorithm C achieves the minimum number of tracks as shown in Fig. 4.15. For this layout pattern, the trunk-crossing count is 7 and the track count is 11. Hence, trunks must be divided in order to reduce the width. Algorithm B first examines matching conditions for max groups.

$d_w(R_1'; G_1 \cup G_2) - W_0(R_1') = 1$,

$d_w(R_1'; G_2 \cup G_3) - W_0(R_1') = 1$,

$d_w(R_1'; G_3 \cup G_4) - W_0(R_1') = 2$,

$d_w(R_1'; G_4 \cup G_5) - W_0(R_1') = 1$,

$d_w(R_1'; G_5 \cup G_6) - W_0(R_1') = 1$,

$d_w(R_1'; G_6 \cup G_7) - W_0(R_1') = 1$

$d_w(R'_1;\ G_7 \cup G_8) - W_0(R'_1) = 1.$

Hence, Algorithm B first examines those w-arcs that exist between V_3^1 and V_3^2. Here the w-arc (t_6, t_4) and (t_6, t_{15}) are excluded because $\xi[L(t_6), U(t_4)] = [L(t_6), U(t_{15})] = \emptyset$. Selected w-arcs are (t_2, t_4) and (t_2, t_{15}). Then, for the w-arc (t_2, t_4), the trunk t_5 is the only trunk to be divided, and the optimum dividing pattern of t_5 is $(t_5, 8, I)$. This division reduces the track count by one without increasing the trunk-crossing count. Algorithm B proceeds in this way and finds the layout pattern shown in Fig. 4.16. For this pattern, the track count is equal to the trunk-crossing count, and hence Algorithm B terminates.

CPU times and other data for all the examples are summarized in Table 4.10.

t_i	$T_L(t_i)$	$T_R(t_i)$	$U(t_i)$	$L(t_i)$
t_1	{1}	{5}	0	5
t_2	{7}	{1, 3}	0	7
t_3	{9, 12}	{7}	7	12
t_4	{10, 15}	{12}	10	15
t_5	{3, 6, 8}	{10, 16}	3	∞
t_6	{2, 5}	{8, 9}	0	9
t_7	{11}	{2}	0	11
t_8	{16, 18}	{11}	11	∞
t_9	$\emptyset$	{14, 18}	14	∞
t_{10}	{4, 13}	{6}	4	13
t_{11}	$\emptyset$	{4, 15}	4	15
t_{12}	$\emptyset$	{13, 17}	13	∞

Table 4.4. Example 1.

y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$t_L(y)$	t_1	t_6	t_5	t_{10}	t_6	t_5	t_2	t_5	t_3	t_4	t_7	t_3	t_{10}	-	t_4	t_8	-	t_8
$t_R(y)$	t_2	t_7	t_2	t_{11}	t_1	t_{10}	t_3	t_6	t_6	t_5	t_8	t_4	t_{12}	t_9	t_{11}	t_5	t_{12}	t_9
n(y)	4	4	5	7	7	6	7	6	6	6	7	6	6	6	6	4	4	4

Table 4.5. Terminal conditions for Example 1.

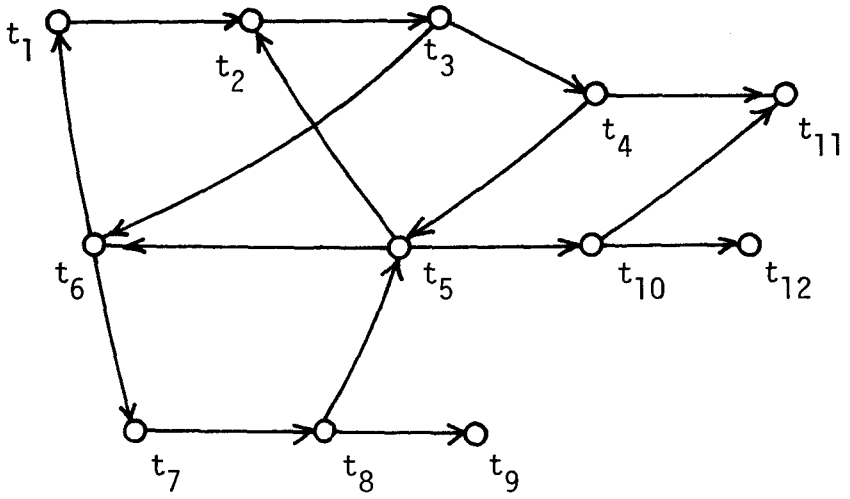


Fig. 4.11. HC-graph for Example 1.

DIVIDING PATTERNS

TRUNK	D-POINT	METHOD	MERIT
2	1	4	?
2	3	4	?
3	9	5	?
5	6	1	?
5	14	1	?
5	10	4	?
5	8	5	?
6	8	4	?
6	9	4	?
4	17	3	?
4	15	5	?
8	14	?	?
8	18	5	?

MAXIMUM NUMBER OF CYCLES TO BE OPENED = 2

Table 4.6. Dividing patterns that eliminate the maximum number of cycles.

MAXIMUM DIVIDING PATTERNS

TRUNK	D-POINT	METHOD	MERIT
2	1	4	6
2	3	4	0
3	9	5	4
5	6	1	7
5	14	1	6
5	10	4	4
5	8	5	4
6	8	4	4
6	9	4	11
4	17	3	40
4	15	5	25
8	14	2	6
8	18	5	6

Table 4.7. $f_3 + f_4$ values for those dividing patterns listed in Table 4.6.

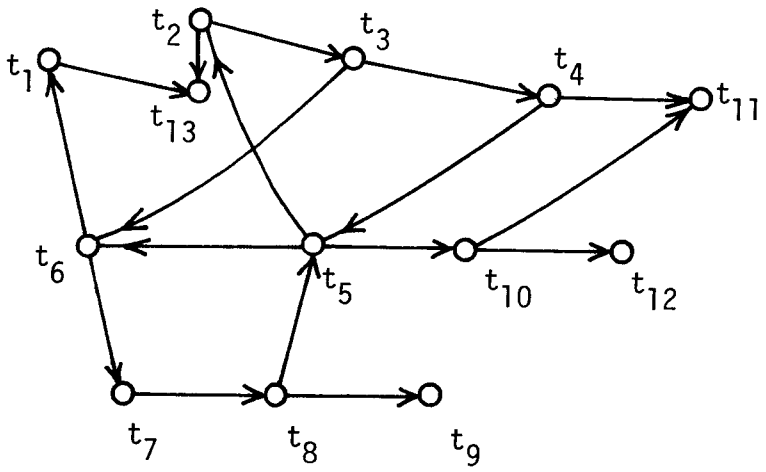


Fig. 4.12. Resulting HC-graph.

MAXIMUM DIVIDING PATTERNS

TRUNK	D-POINT	METHOD	MERIT
3	9	2	907
3	9	5	904
4	17	3	840
4	15	5	825
5	6	1	907
5	8	1	907
5	8	5	904

Table 4.8. $100f_2 + f_3 + f_4$ values for those dividing patterns which eliminate the remaining cycle.

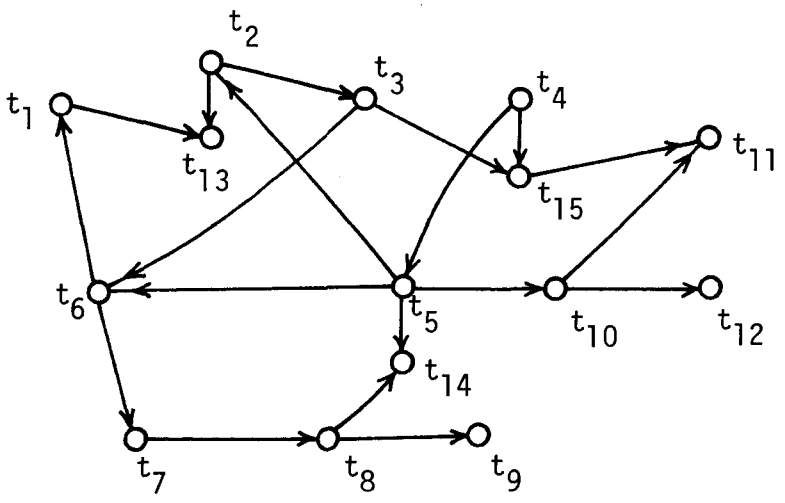


Fig. 4.13. Resulting acyclic HC-graph.

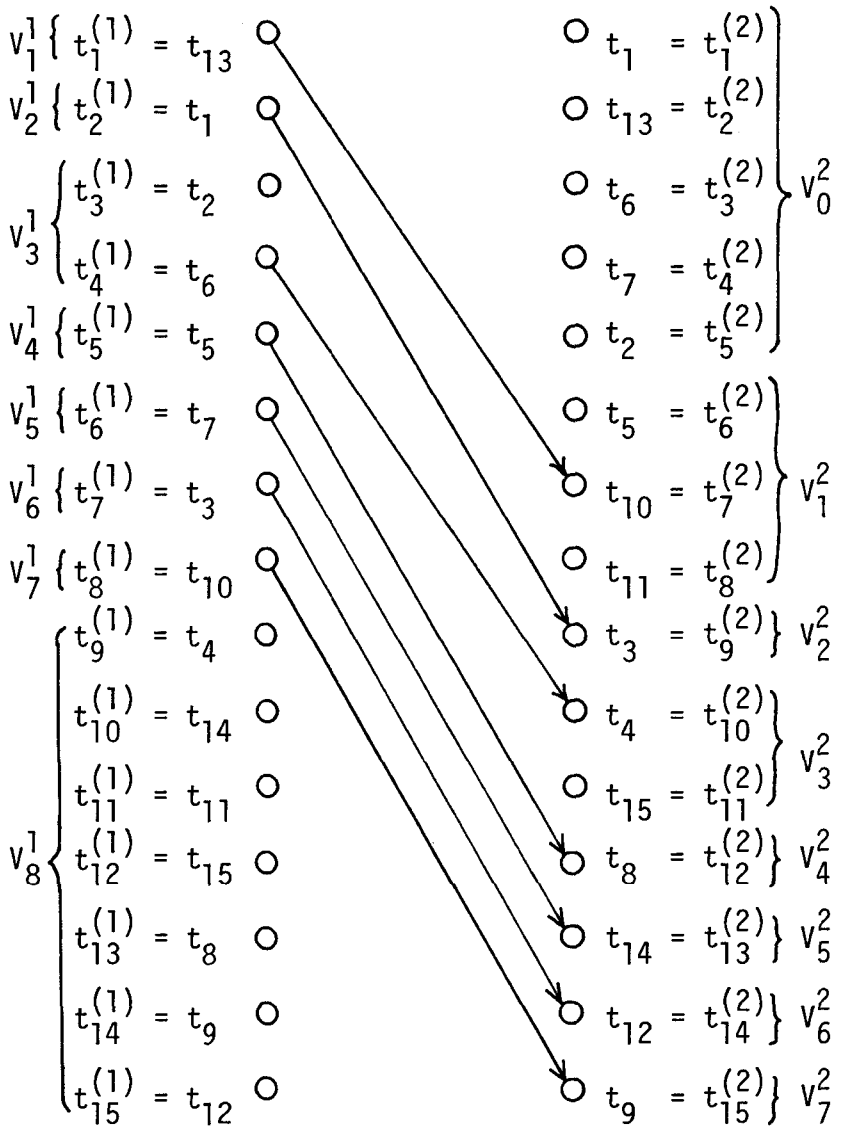


Fig. 4.14. Simplified VR-graph.

t_i	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
$x_L(t_i)$	6	3	4	1	2	5	6	7	8	3	6	4	7	5	8
$x_R(t_i)$	2	7	6	9	8	5	4	3	1	3	1	1	1	2	1

Table 4.9. x_L and x_R values.

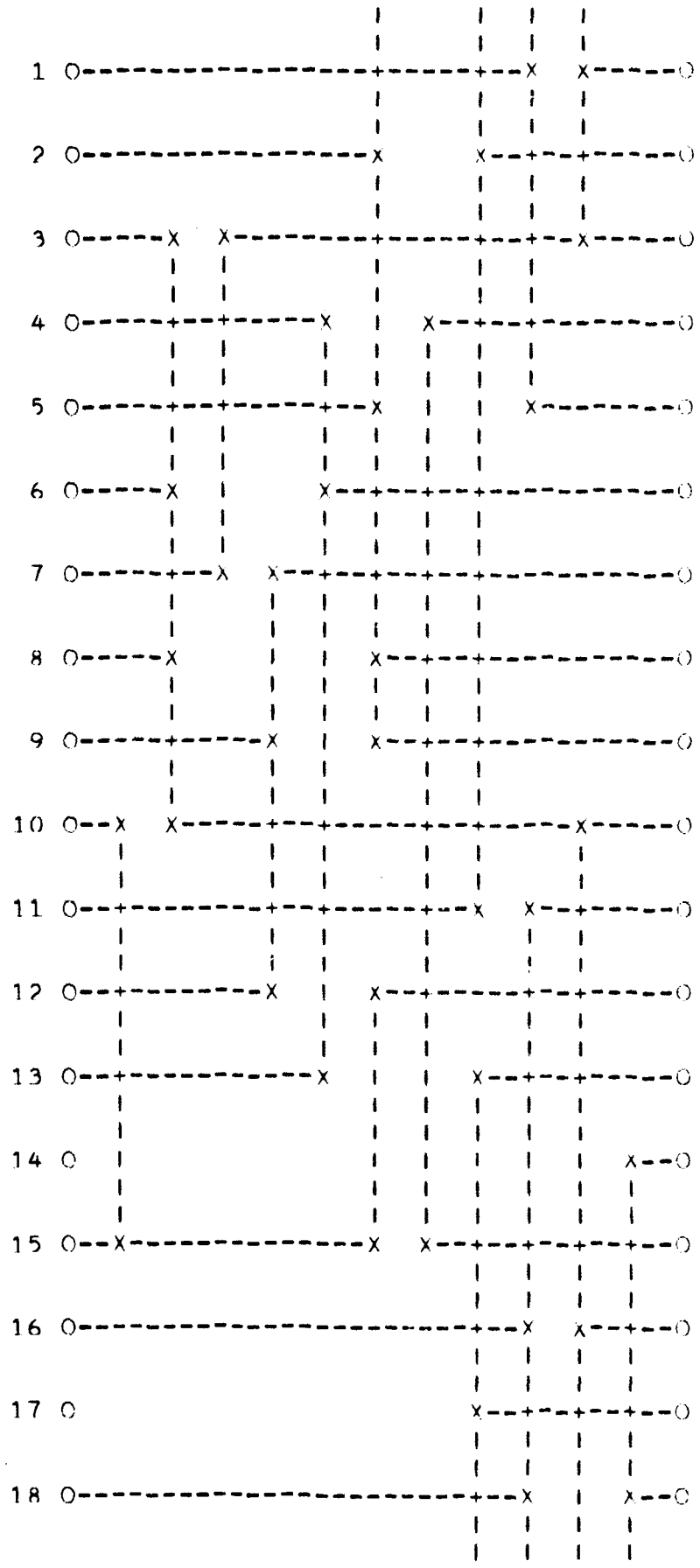


Fig. 4.15. Layout pattern of width 11 from which Algorithm B starts.

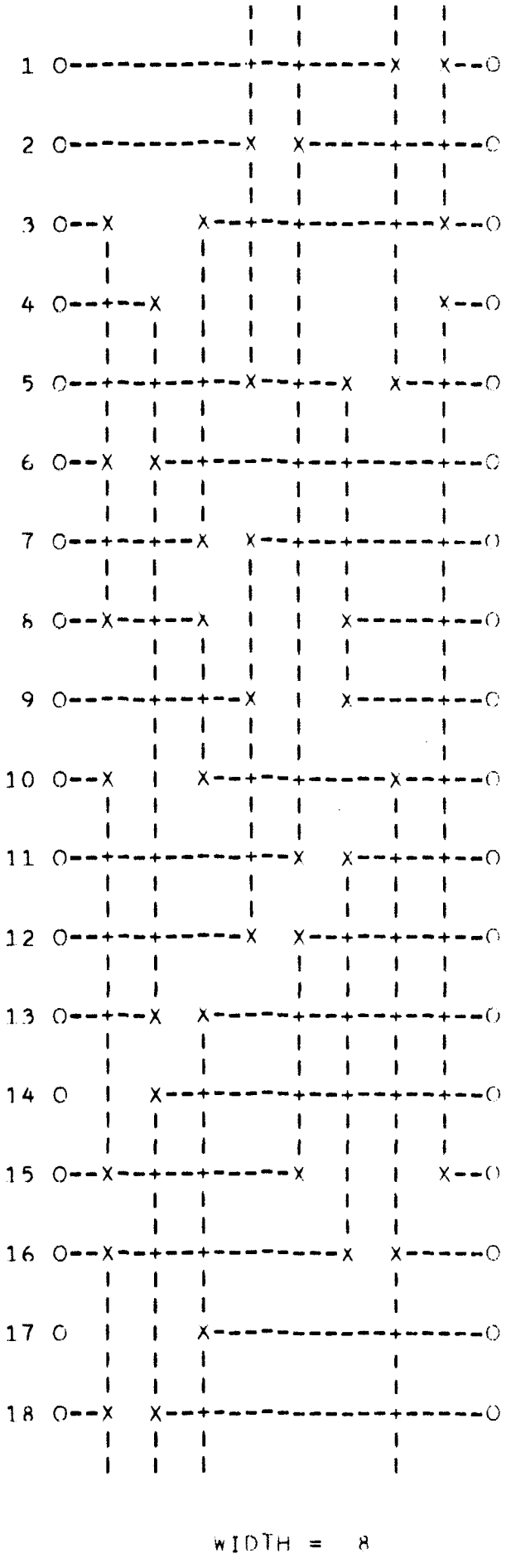


Fig. 4.16. Final layout pattern of width 8.

Example Number	Number of Wires	Number of Gates	Number of Cycles	$W_0(R)$	Initial Width	Final Width	Number of Divided Trunks	Total Execution Time (sec)
1	12	18	5	7	11	8	6	4.6
2	5	7	3	4	7	6	3	0.9
3	10	12	3	7	10	9	3	2.7
4	10	25	65	10	14	12	8	42.3
5	12	16	1	10	10	10	1	0.8
6	12	16	9	11	15	14	7	8.8
7	12	17	0	5	6	6	0	0.5
8	13	20	0	5	6	5	1	1.9
9	15	22	2	12	12	12	1	1.6

Table 4.10. Computational results.

4.9 Conclusions

The structure of the width reduction algorithm (Algorithm B) described in this chapter is more complicated than that of the cycle eliminating algorithm (Algorithm A) presented in the previous chapter. This complexity of the structure is due to the difficulty of evaluating dividing patterns of trunks, and the difficulty is explained by the fact that Algorithm A checks only whether a dividing pattern eliminates cyclic constraints in an HC-graph, but Algorithm B must examine with the tedious work of rearranging trunks whether a division can really reduce the track count. Thus, it is the most important in Algorithm B to limit the search space. For this purpose various notions and conditions have been introduced such as matching conditions, incomplete subsets and so on. Lastly, this approach has proven successful by means of experimental results.

CHAPTER 5

CONCLUSIONS

The wire-routing design problem considered in this thesis has been studied in many parts of the world, and toward this problem various kinds of approaches have been reported. However, none of them could be fully successful for lack of the ability to achieve perfect wirability. Imperfect wirability may be considered to be a serious defect because of inevitableness of human aids in adding incompleted wiring routes. In this reason it is not too much to say that the algorithm presented in this thesis is epoch-making, which achieves perfect wirability. The success of this approach depends greatly upon the mathematical model of the wire-routing system and the trunk-division methods. This thesis evaluates the operation of dividing a trunk from two viewpoints, (1) its ability to eliminate cyclic constraints and (2) its ability to cut off long constraint chains in an HC-graph. Then, by the effective use of those trunk-division methods the author could have an excellent wire-routing algorithm. Also, the experimental results have proven the validity and the usefulness of the algorithm.

The only work left for further improvement is to achieve better element placing. The current version of the wire-routing program is independent of a program for placing elements in a suitable order. However, these programs, in themselves, should work in close cooperation with each other. The author will study such a cooperating-program system.

LIST OF REFERENCES

1. Luccio, F., and M. Sami, On the Decomposition of Networks in Minimally Interconnected Networks, IEEE Trans. on Circuit Theory, Vol. CT-16, No. 3, pp. 184-188, May 1969.

2. Landman, B. S., and R. L. Russo, On a Pin Versus Block Relationship for Partitions of Logic Graphs, IEEE Trans. on Computers, Vol. C-20, No. 12, pp. 1469-1479, December 1971.

3. Russo, R. L., and P. K. Wolff, A Computer-Based-Design Approach to Partitioning and Mapping of Computer Logic Graphs, Proc. IEEE, Vol. 60, No. 1, pp. 28-34, January 1972.

4. Russo, R. L., Oden, P. H., and P. K. Wolff, A Heuristic Procedure for the Partitioning and Mapping of Computer Logic Graphs, IEEE Trans. on Computers, Vol. C-21, No. 12, pp. 1455-1462, December 1972.

5. Steinberg, L., The Backboard Wiring Problem: A Placement Algorithm, SIAM Review, Vol. 3, No. 1, pp. 37-50, January 1961.

6. Rutman, R. A., An Algorithm for Placement of Interconnected Elements Based on Minimum Wire Length, Proc. SJCC, pp. 477-491, 1964.

7. Fisk, C. J., Caskey, D. L., and L. E. West, ACCEL Automated Circuit Card Etching Layout, Proc. IEEE, Vol. 55, No. 11, pp. 1971-1982, November 1967.

8. Prim, R. C., Shortest Connection Networks and Some Generalizations, Bell Syst. Tech. J., Vol. 36, No. 11, pp. 1389-1401, November 1956.

9. Loberman, H., and A. Weinberger, Formal Procedure for Connecting Terminals with Minimum Total Wire Length, JACM, Vol. 4, No. 4, pp. 428-433, October 1957.

10. Moore, E. F., Shortest Path Through a Maze, Annals of the Computation Laboratory of Harvard University, Vol. 30, pp. 285-292, 1959.

11. Lee, C. Y., An Algorithm for Path Connections and its Applications, IRE Trans. on Electronic Computers, Vol. EC-10, No. 5, pp. 346-365, September 1961.

12. Heiss, S. A., A Path Connection Algorithm for Multi-Layer Boards, Proc. Design Automation Workshop, pp. 1-12, 1968.

13. Hightower, D. W., A Solution to Line-Routing Problems on the Continuous Plane, Proc. Design Automation Workshop, pp. 1-24, 1969.

14. Mikami, K., and K. Tabuchi, A Computer Program for Optimal Routing of Printed Circuit Conductors, Proc. IFIP, Vol. 2, pp. 1475-1477, 1968.

15. Hitchcock, R., Cellular Wiring and the Cellular Modelling Technique, Proc. Design Automation Workshop, pp. 25-41, 1969.

16. Lass, S. E., Automated Printed Circuit Routing with a Stepping Aperture, CACM, Vol. 12, No. 5, pp. 262-265, May 1969.

17. Hashimoto, A., and J. Stevens, Wire Routing by Optimizing Channel Assignment within Large Apertures, Proc. Design Automation Workshop, pp. 155-169, 1971.

18. Akers, S. E., A Modification of Lee's Path Connection Algorithm, IEEE Trans. on Electronic Computers, Vol. EC-16, No. 1, pp. 97-98, January 1967.

19. Kernighan, B. W., Schweikert, D. G., and G. Persky, An Optimum Channel-Routing Algorithm for Polycell Layouts of Integrated Circuits, Proc. Design Automation Workshop, pp. 50-59, 1973.

20. Asano, T., Kitahashi, T., Tanaka, K., Horino, H., and T. Amano, Realizability of Wiring for Building-Block Type LSI, Trans. IECE,

Vol. 56-A, No. 9, pp. 489-496, September 1973 (in Japanese).

21. Asano, T., Kitahashi, T., Tanaka, K., Horino, H., and T. Amano, A Graph Theoretical Approach to the Routing Problem, Trans. IECE, Vol. 56-A, No. 12, pp. 731-738, December 1973 (in Japanese).

22. Liu, C. Y., Introduction to Combinatorial Mathematics, McGraw-Hill, L968.

23. Asano, T., Kitahashi, T., and K. Tanaka, On a Method of Realizing Minimum-Width Wirings, Trans. IECE, Vol. 59-A, No. 2, pp. 115-124, February 1976 (in Japanese).

24. Asano, T., Kitahashi, T., Tanaka, K., Horino, H., and T. Amano, A Wire Routing Scheme Based on Trunk Division Methods, IEEE Trans. on Computers (to appear).