AN APPROACH TO MCM ROUTING IN MANHATTAN DIAGONAL MODEL

A dissertation submitted in partial fulfillment of the requirement of M. Tech. (Computer Science) Degree of Indian Statistical Institute, Kolkata
by

Amit Kumar Banerjee

Under the Supervision of
Dr. Sandip Das &
Dr. Subhas C. Nandy

Advanced Computing and Microelectronics Unit,
Indian Statistical Institute, Kolkata

Indian Statistical Institute, Kolkata
203, Barrackpore trunk road,
Kolkata – 700108

19th July, 2002

Indian Statistical Institute

203, Barrackpore Trunk road,
Kolkata - 700108

Certificate of approval

This is to certify that this thesis titled with **"An approach to MCM routing in Manhattan Diagonal model"** submitted by **Amit Kumar Banerjee** submitted toward partial fulfillment of requirements for the degree of M. Tech. In Computer Science at Indian Statistical Institute, Kolkata embodies the work done under my supervision.

Sandip Das

Dr. Sandip Das
Dr. Subhas C. Nandy
Advanced Computing and Microelectronics Unit,
Indian Statistical Institute
Kolkata - 700108

Acknowledgement

I take pleasure in thanking Dr. sandip Das and Dr. Subhas C. Nandy for there friendly guidance throughout the dissertation period. Their pleasant and encouraging words has always kept my sprit up.

I take my opportunity to thank my classmates, friends and my family for their encouragement to finish this work.

Amit Kumar Banerjee

Amit Kumar Banerjee

Contents

Chapter 1

Introduction

With the rapid increase of feature size, efficiently packing multiple circuit modules in a single chip has become a promising area of research. Using MCM technology a large number of functional blocks can be mounted and interconnected in a single substrate. The number of distinct nets and terminals are usually very high. A typical MCM may consist of 7000 nets on a grid size of over 2000 X 2000 [2]. With the emergence of deep sub micron technology the number of chip, nets and terminals per net, in an MCM are likely to increase in the future.

Inside an MCM the circuit components are mounted on the top layer. Other layers are used for pin redistribution and routing. As many layers are usually available for MCM routing, the main objective is to improve performance, i.e. to achieve tight packaging with lower fabrication cost, satisfying the constraints on net length, net separation, via size, via separation etc. Thus the primary goal is to minimize the number of vias, wire length, cross talks, delay in signal propagation, with the secondary objective of reducing the number of routing layers in the MCM.

Several approaches to MCM routing have been proposed recently [4]. A pair of routing layers called the layer-pair is considered at a time and yet unrouted terminals are projected on them though stack vias. Khoo and Cong developed SLICE [2] and VAR [3]. In the latter method at most four vias are used for routing two terminal net in a non-monotone fashion. Multi-terminal nets are transformed into two-terminal nets. Another algorithm, called SEGRA, was proposed in [1]. It uses a plane sweep technique and splits each multi terminal net into two-terminal nets considering a minimum spanning tree among its terminals.

This algorithm outlines a new approach to MCM routing in Manhattan-diagonal model, that combines the advantage of both diagonal model and Manhattan geometry. Here two routing layers with fixed terminals in the Manhattan diagonal ($\pm 45^{o}$) model has been considered. Thus, the wire segments may either be rectilinear or at $\pm 45^{o}$ directions. The diagonal model is well supported by the fabrication process and provides better utilization of routing space, tends to reduce wire length and hence area and delay. However the existing algorithms based on the model are either too elementary or inefficient and therefore have limited capability to real life routing problems.

Non-overlap model is considered in this routing algorithm. In non-overlap model generally those algorithms, which assume only vertical and horizontal wire segments, put all the non-overlapping vertical segments in a layer and all non-overlapping horizontal wire segments in the other layer of a layer-pair. But in this model, all the non-overlapping horizontal wire segments with some $\pm 45^{o}$ bend are put in a layer and non-overlapping vertical wire segments with some $\pm 45^{o}$ bends are put in the other layer of the layer pair.

Chapter 2

Preliminaries

The MCM routing problem consists of a rectangular multi-layer routing substrate, a set of rectangular circuit modules (dies), and a set of nets. Dies are mounted on the top layer of the routing substrate, and the other layers are used for pin redistribution and routing. The die terminals are attached to the substrate by using some bonding technology. The objective is to interconnect all the terminals of same signal net such that there is no overlap of wiring with that of other nets in the same layer.

Definition: 3-D routing space is a three dimensional space divided into a number of layers parallel to X-Y plane. The layers are numbered from top to bottom, and the top layer is marked as *layer-0*. Each layer consists of a finite number of grid points generally by a set of horizontal and vertical tracks. A grid point is defined by (x, y, z) in 3-D space, where x and y coordinates have there usual meaning and z coordinate indicates the layer identification. A line parallel to the X-axis (Y-axis) is referred as horizontal (vertical) line and a line in the $\pm 45^{o}$ direction is called as diagonal line.

A net, consisting of a set of terminals $t_1, t_2 \ldots , t_k,$ are attached to the boundaries different of circuit modules, which are to be interconnected. The rectangular circuit modules are arranged on the substrate in the form of a K x L mesh of rectangles on *layer – 0* of the 3-D routing space. A pin redistribution layer is often needed when the spacing between pins do not satisfy the design rule for routing. The grid lines are assumed to be equally spaced and the routing pitch for the given technology determines the spacing. In the discussion pin redistribution layer is not considered as it is assumed that the pad spacing (spacing between the terminals) satisfies the design rule for routing. A scheme of the model is shown in Fig. 1. It is assumed that the circuit modules are identical in size and shape, terminals are equally spaced, and no terminal is allowed to be routed through them. The terminals, marked with '0' represent blank terminals.

The router routes two adjacent layers at a time. Usually the vertical and horizontal connections are made in two different layers (non-overlap). But as discussed earlier we allow some diagonal wire segments in both horizontal and vertical layers.

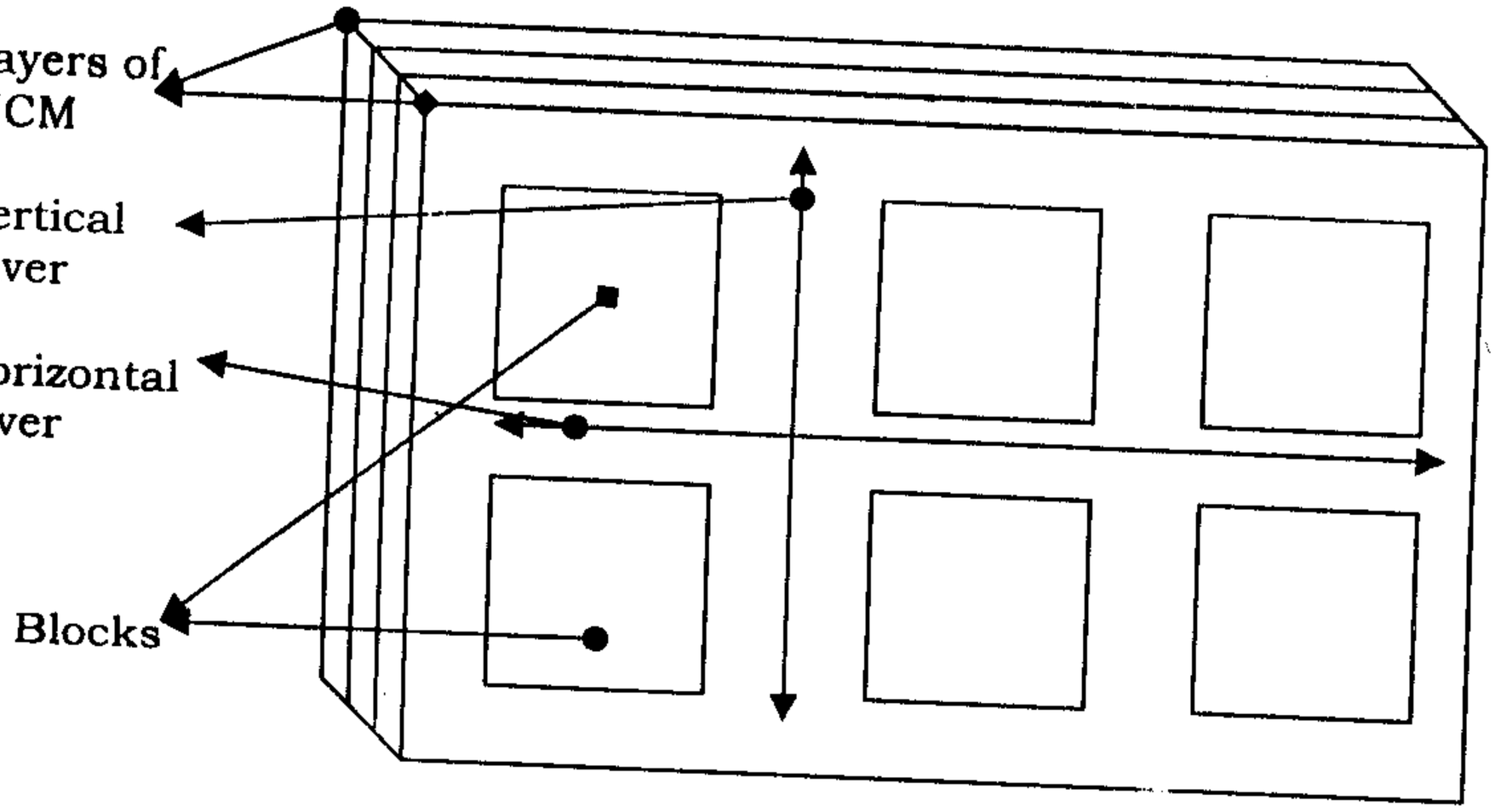


Fig. 1: Schematic description of an MCM

Fig. 2 demonstrates the classification of different type of vias. An *interconnection via* or simply via is a hole through which a connection is made to the adjacent layers. Vias may be stacked on top of each other to form a *stack via* for connecting wires in non-adjacent layers. A stack via contributes k to *via count* if it penetrates k layers. In this model, the stack vias are placed at the original terminal positions of the nets that are in *layer – 0*. The vias used for propagating the die terminals to the first routing layer are called *distribution via*. These vias do not contribute to via count.

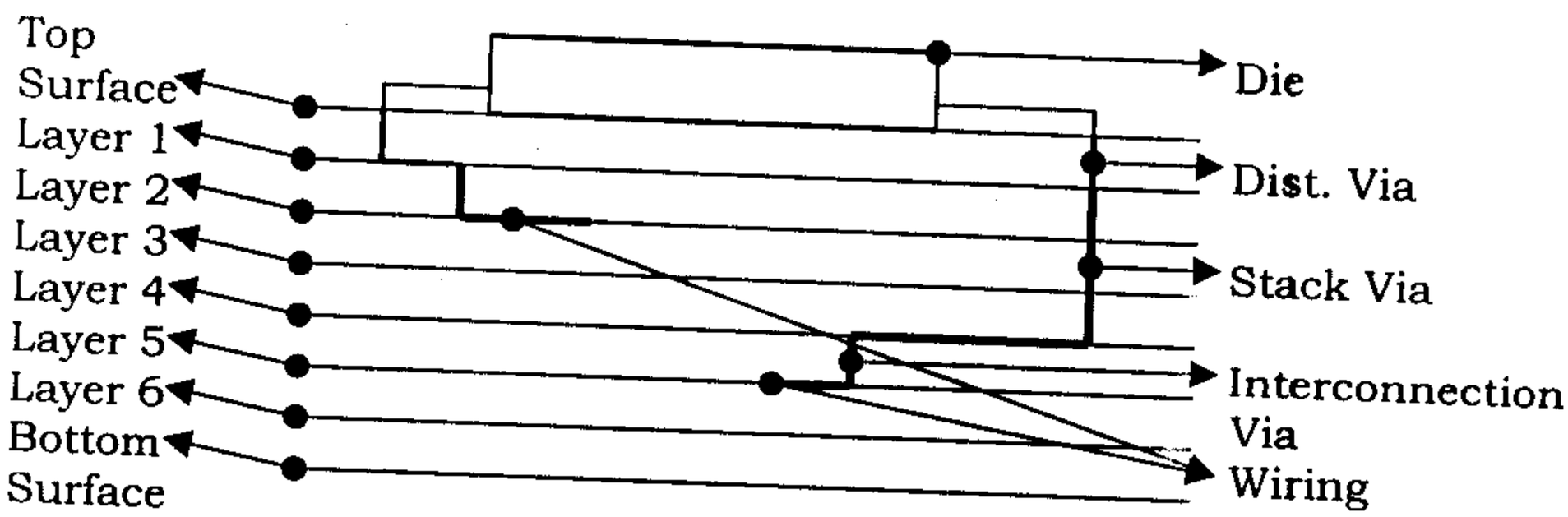


Fig. 2: Classification of via

Definition: The horizontal (vertical) routing channels between each pair of consecutive rows (columns) of blocks are termed as *X - river* (*Y – river*). In a K x L mesh of blocks we denote the *X – rivers* as R_1, R_2,....., R_{K+1}, and the *Y – rivers* as C_1, C_2, ..., C_{L+1}.

If the four sides of a block are projected on a routing layer the rectangular space bounded by these lines is termed as *space inside the block*. All *X – rivers* and *Y – rivers* as well as the space inside the blocks are used for routing in this algorithm. Sometimes, layer area may need to be increased to achieve routability.

Chapter 3

Overview of the algorithm

Definition: The *Steiner Minimum Tree* (SMT) is defined as follows: Given an edge weighted graph $G = (V, E)$ and a subset $D \subseteq V$, select a subset V' $\subseteq V$, such that $D \subseteq V'$ and V' induces a tree of minimum cost over all such trees. The set D is referred to as the set of *demand points* and the set $V' - D$ is referred to as *Steiner points*. If $D = V$ then SMT is equivalent to *Minimum Spanning Tree* (MST). Finding SMT is a NP-complete problem. In view of the NP-completeness of the problem several heuristic algorithms have been developed [6].

In the routing problem, we consider all edges are either horizontal or vertical. So the set of edges in the graph are restricted to be rectilinear.

Definition: A Steiner tree whose edges are constrained to be rectilinear, is called a *Rectilinear Steiner Tree* (RST). A *Rectilinear Steiner Minimum Tree* (RSMT) is an RST of minimum cost among all RSTs.

If the set V is defined as the set of blank grid points at any intermediate stage of routing in the corresponding routing layer, the set E is defined as all possible rectilinear edges among the nodes in V which do not use any of the non-blank grid points, the set D (*demand points*) is defined as the terminal grid points of a particular net a_i and the edge weights are defined as the length of the edge then the RSMT gives the routing edge with minimum wire length. But this RSMT does not ensure the minimum number of bends between horizontal and vertical paths. As each bend causes a via, it does not give the minimum number of vias on the path. Fig. 3 gives an example.

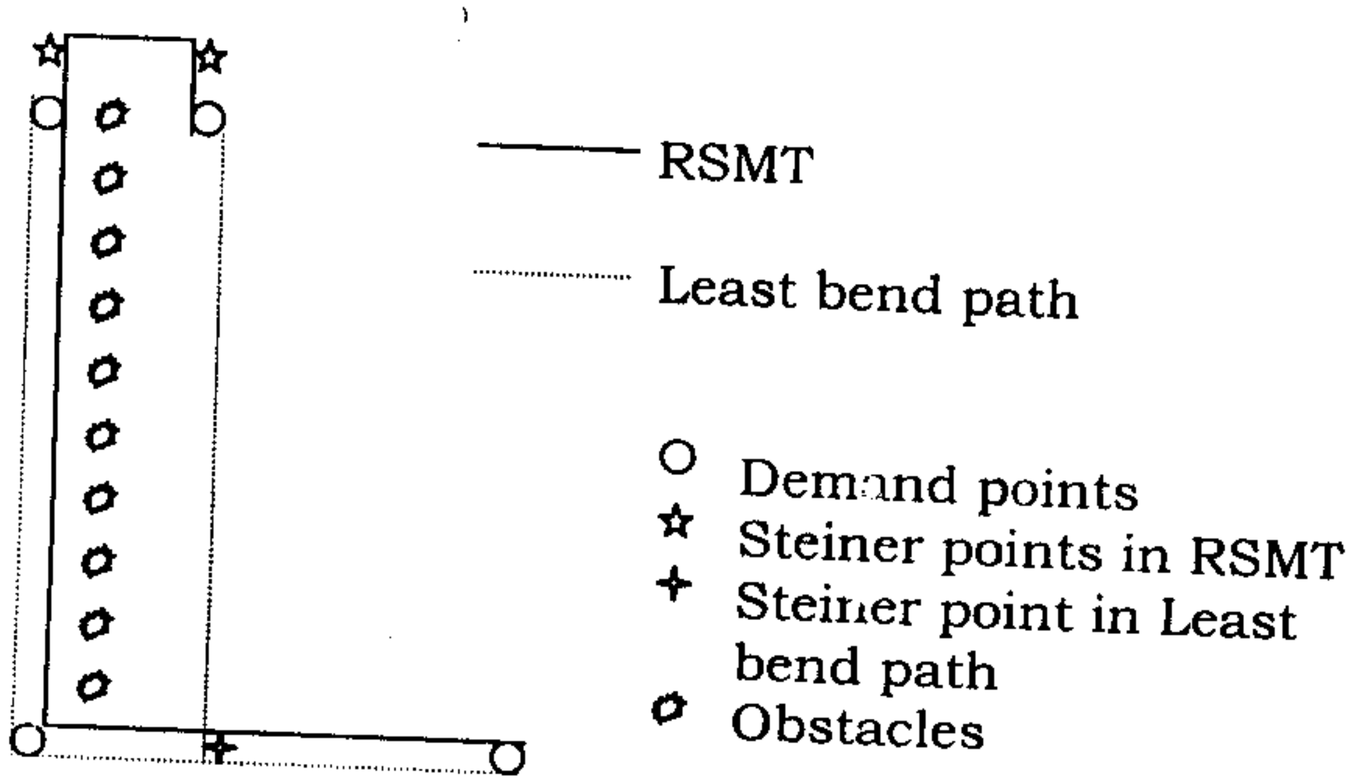


Fig. 3: RSMT and least bend path for a set of demand of demand points

In our algorithm we do the following steps:

- The set V is defined as the set of all grid points including the non-blank grid points, the set E is defined as all-possible rectilinear paths among the nodes in set V, the set D (*demand points*) is defined as the terminal grid points of a particular net a_i and the edge weights are defined as the length of the path, then the RSMT gives the routing path which has minimum number of bend and also with minimum wire length.

- But this RSMT may not give a feasible path as it may contain some grid points, which belong to some other RSMT or some terminal grid point of some other net. This RSMT contains only horizontal and vertical paths. Now, diagonal paths are used to shift the horizontal path to upper or lower direction and vertical paths to left or right side if the paths come across an obstacle (i.e. grid points which belong to some other RSMT or some terminal grid point of some other net). This is illustrated in figure 4.

- Every horizontal or vertical edge routing is started from a terminal point of the corresponding net and it falls on a vertical or horizontal edge (hereafter will be called as *target edge*), so shifting does not introduce a new edge, instead it only increases total wire length. With every left or right shifting of vertical path (top or bottom shifting of horizontal path) may cause increment of target horizontal path to the left or right (target vertical path to the top or bottom) direction. Thus shifting of horizontal or vertical paths are done as minimum as possible to keep the increment of path length minimum. But as no extra edge is introduced, the number of vias required to rout the net in non-overlapping model is the lowest possible. Increment of target edges are taken care of by the algorithm.

- These all non-overlapping horizontal paths (may be shifted to left or right) with diagonally bends are kept in one layer and similarly all non-overlapping vertical paths (may be shifted to left or right) with diagonally bends are kept in the other layer of a layer-pair.

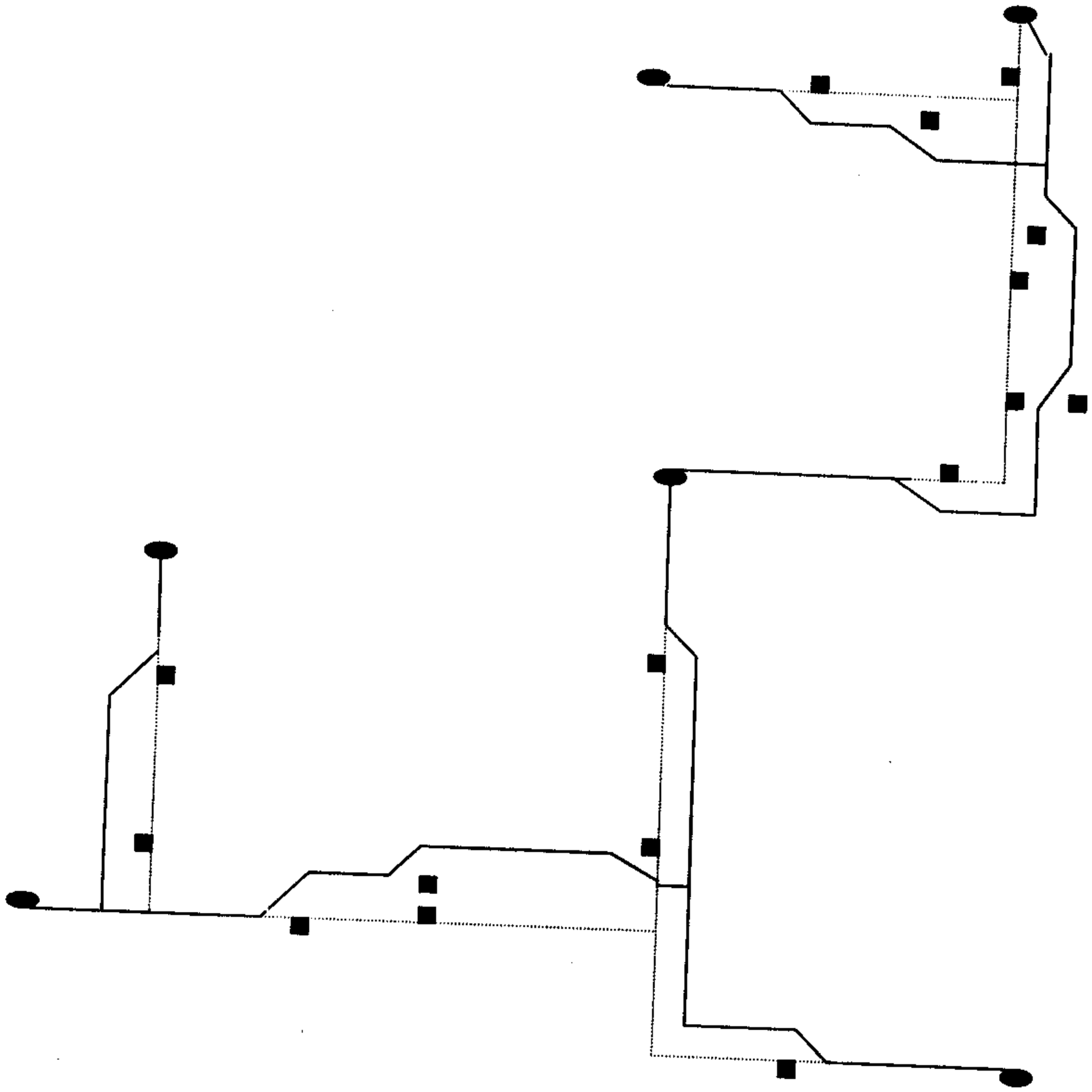


Fig. 4 : Original RSMT and actual routing

Chapter 4

Proposed routing technique

Let a_1, a_2, ..., a_m be m distinct nets whose terminals are attached along the boundaries of different blocks as well as the floor. The number of terminals for net a_i is n_i where $n_i \geq 2$ for all $i = 1,....., m$. Initially the terminals of all nets are available at *layer – 0* along the boundaries of the blocks. These terminals are propagated to their appropriate routing layers using stack vias. In the following section the algorithm is described.

4.1 Determination of RSMT

In an intermediate routing layer, we compute the RSMT of all the yet unrouted nets a_i, a_{i+1}, ..., a_j RSMT is determined following some well-known algorithm[6].

After actual routing, the edges are shifted by diagonal bends. The horizontal edges with necessary $\pm 45^o$ bends are supposed to be routed in the one layer and all the vertical edges with necessary $\pm 45^o$ bends are supposed to be routed in the other layer of the layer pair. But as the space between the blocks (river width) and inside the blocks is finite, some nets may not be routed in this layer. They may be projected to the next layer through stack vias.

4.2 Routing of nets

Routing of a net is done separately in horizontal and vertical layer following similar method. In order to facilitate the routing, we define the followings.

Definition: The horizontal layer (vertical layer) is divided into two nearly equal parts (with respect to the number of blocks) with a vertical (horizontal) channel. The parts to the left and right (top and bottom) sides of the vertical (horizontal) channel are called *left division* and *right division* (*top division* and *bottom division*) respectively. The strips defining the divisions should be as wide as possible. Fig. 5 illustrates the division.

4.2.1 Phase 1

The routing in *left* and *right* (and hence the *top* and *bottom*) divisions are done independently, without considering the other divisions in this phase. Each horizontal (vertical) edge of the RSMT is assumed to start from a terminal and finishes at a point on another vertical (horizontal)

edge. If the starting point of a horizontal edge lies in the left division and target point is in the right division then in phase 1, we route the portion of the edge up to the boundary of the left division. The first phase of routing inside a division is divided into two parts a) routing inside a block and b) routing outer side of blocks.

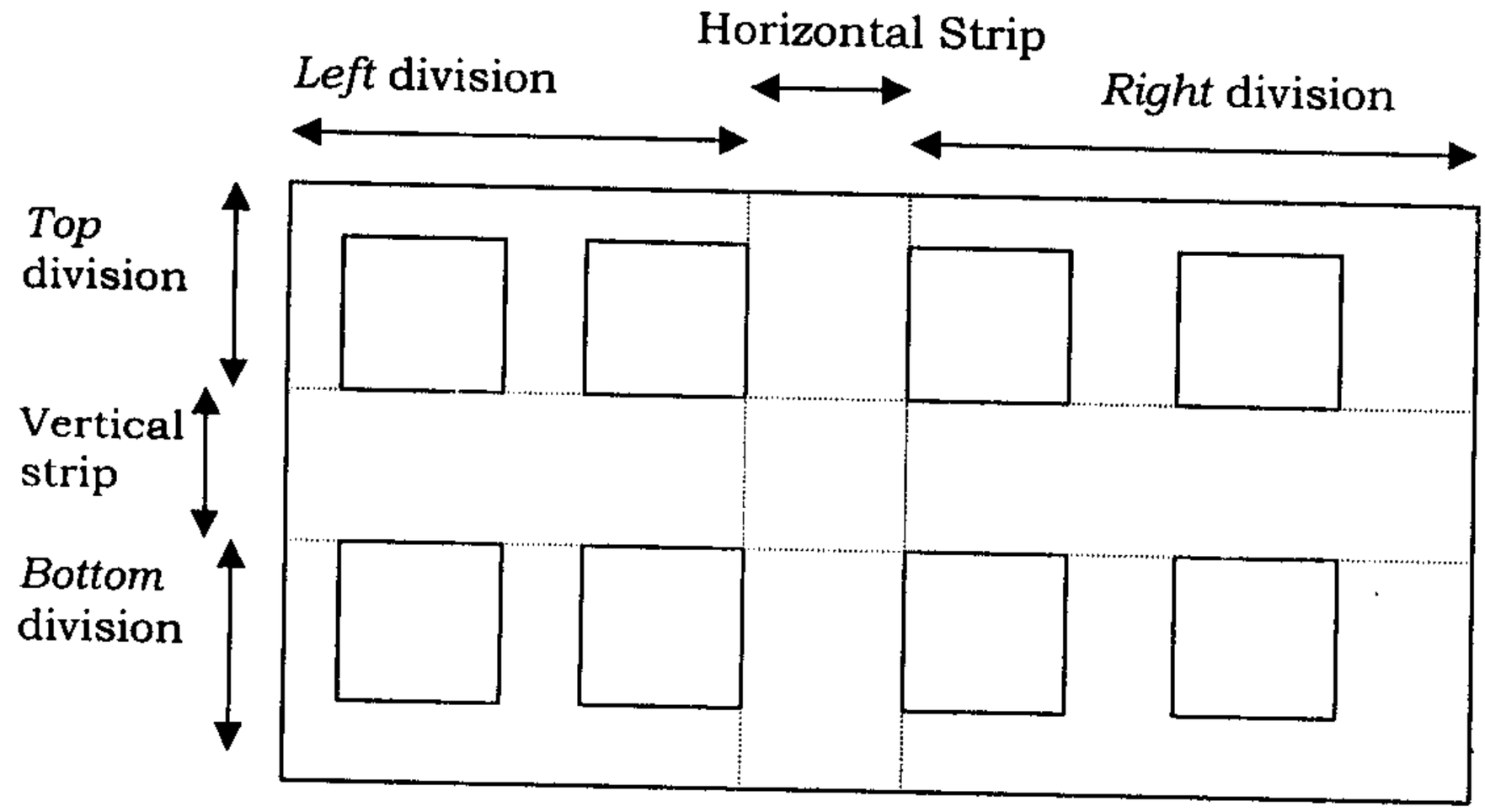


Fig. 5: Horizontal and vertical division

4.2.1.1 Routing inside a block

Routing inside a block is illustrated for only *left* division of a horizontal layer. Other cases (*right* division of horizontal layer, *top* and *bottom* division of vertical layers) are done following the similar way.

A *gap* on the boundary of a block may be defined as consecutive blank terminals on the boundary of a block. A gap may have one or more blank terminals in it. For each block the gaps are determined, and one kept in doubly connected link list.

Horizontal edges

Now the top (bottom) horizontal boundary of a block is searched for the starting of a path corresponding to a net, which has the target on the left side of the starting point. At each block, the top (bottom) boundary is searched from left to right to get the start vertex of an edge which has target point to the left of the start vertex. So each edge is routed up to

the top most available blank terminal of the left boundary as mentioned in the next paragraph. These parts are all parallel. Note that, if the target is at the right side of the left boundary (as in he fig. 6), then also we route it up to the left boundary of the boundary of the block.

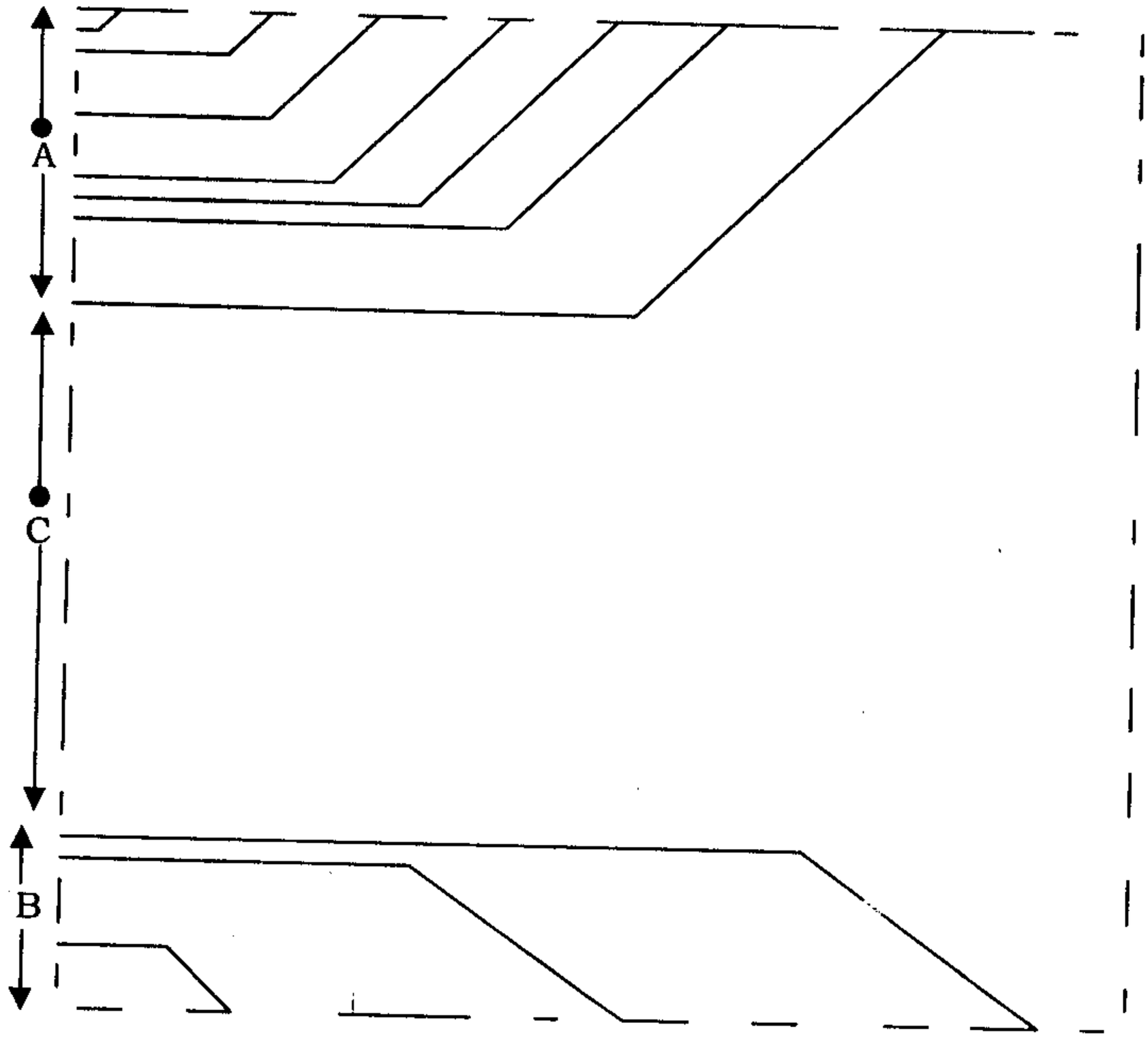


Fig. 6: Horizontal boundary routing

For a path with its starting point at the top boundary of a block, the routing inside a block is done as follows. Select the topmost along the left boundary of the block. If that gap contains only one blank terminal, the path spread up to that terminal and the gap is deleted from the data structure. If more than one terminal is available in that gap, after the routing the gap shrinks i.e. the topmost point of the gap is changed to the point immediately below to this point. Thus inside the block each net is routed using diagonal and horizontal tracks as in Fig. 6.

Left vertical boundary

Now we consider the routing of the edges, which have its starting points at the left boundary of the block and target at the right side. The left

boundary is divided into three parts, a) the *top* part: this part starts at the topmost terminal of the left boundary and spans up to the terminal of the left boundary through which the routing of the nets at the top boundary of the same block ends (see the region marked A in Figure 6), b) the *bottom* part: this part starts from the lowest terminal of the left boundary and spans up to the terminal through which the routing of the nets at the bottom boundary of the block ends (see the region B in Figure 6), c) the *middle* part: it is the portion between the *top* and *bottom* parts (region C in Figure 6).

All the paths, which have its starting points at the *top* (*bottom*) part of left boundary of the block, and target at the right side of the starting point of that edge bare searched starting from the topmost (bottommost) point. As soon as such an edge is found, it is routed (using horizontal and diagonal segments) through a blank terminal of the gap (henceforth will be referred as *right-most reach* of the path) of upper (lower) boundary if such a gap exists. The assignment of gaps is done in the left to right order. At each choice of a blank terminal, the gap structure is updated. Note that such a routing ensures no crossing among the routing of different edges. If such a gap point is not available then usually we abort such a routing excepting the case where the edge has target is to the left of the rightmost reach of the path. In this case, it is routed to the rightmost reach (it is not a blank terminal). See Figure 7 for demonstration.

Paths starting from *middle* part of left edge are joined to the available gaps on the right edge with minimum possible shifting. Suitable paths are searched from *middle* part starting from topmost point. As soon as a path is found which originates from a terminal on the left boundary, we route it up to a blank terminal of right boundary, which has the least vertical deviation with the starting point. Here also, we may need short diagonal segments in addition to the horizontal segments for the routing.

Right boundary

Edges starting from the right boundary of a block and have target towards the left direction are searched from the topmost points to the bottommost point of the right boundary. As soon as a path is found, points in the gap on the left boundary are searched through which it can be joined without collision. From the available points the topmost point is selected, and routed with horizontal and diagonal wire segments. Instantaneously the gap structure is updated.

Now the routing inside the block is over. In the next stage of routing of the left division blocks, we consider only the following points.

A) All the points on the top or bottom boundary, which have their target at the right side of it. Some of these points may be starting point of a path and others are the intermediate points of a path.
B) All the points on the left boundary, which have their target at the left side of it. Some of these points may be starting point of a path and others are the intermediate points of a path.
C) All the points on the right boundary, which have their target at the right side of it. Some of these points may be starting point of a path and others are the intermediate points of a path.

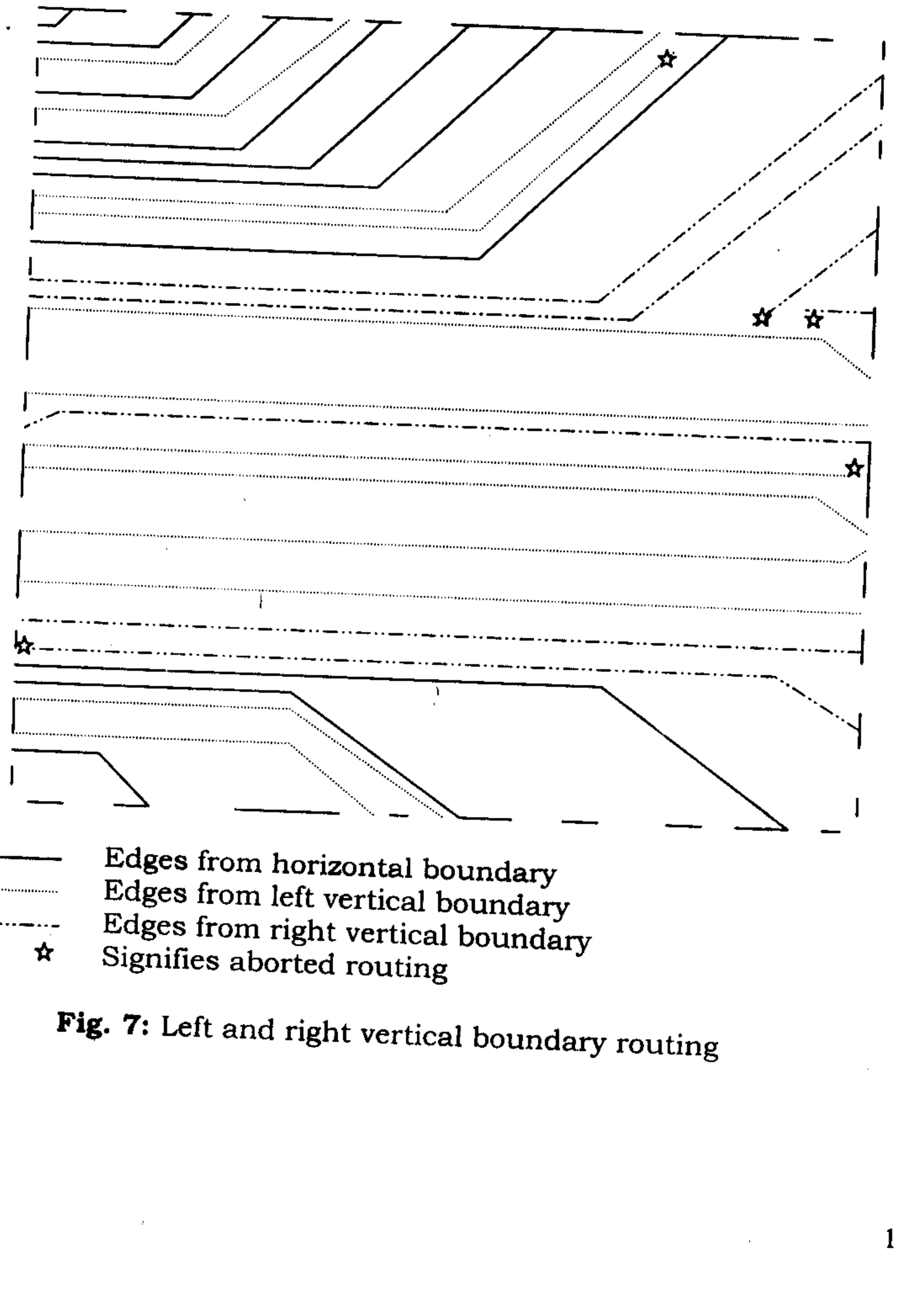


Fig. 7: Left and right vertical boundary routing

4.2.1.2 Routing outer side of blocks in a division

Here this part is illustrated for *left* division of the horizontal layer only. Other cases (*right* division of horizontal layer, *top* and *bottom* division of vertical layer) are done similarly.

Horizontal sides of a block

First a block is considered. Points from the upper (lower) boundary, which have target towards the right direction, are searched from right to left. They are put in the adjoining horizontal river one after one according to the figure Fig. 8. In this stage all edges are routed up to the rightmost X-coordinate of the corresponding block. Fig.8 illustrates the routing.

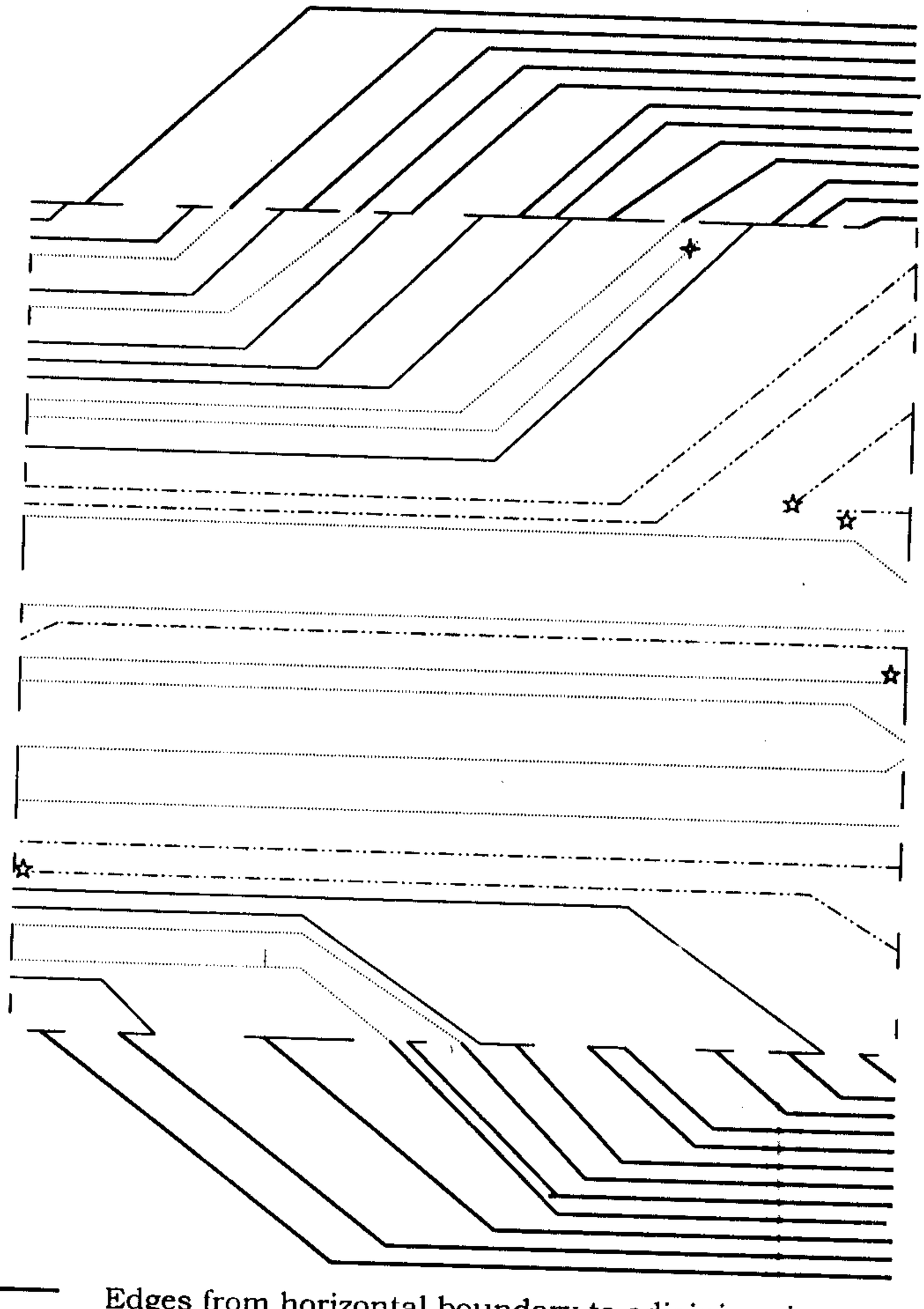
Vertical sides of blocks

After this stage all the blocks pairs in the left division of horizontal layer are considered which are side by side and share a common vertical river in between them. These pairs are considered starting from leftmost vertical river to the rightmost in the division. In a river the block-pairs are chosen starting from the topmost to the bottommost.

In a block-pair, points, which have target to the right direction, are considered at the right vertical boundary of the left block. These two vertical boundaries (right vertical boundary of the left block and left vertical boundary of the right block) are divided into two equal parts, *top* and *bottom* (as in figure 9). Points on this boundary are considered starting from middle point to the extreme point of the *top* (*bottom*) part of the left boundary. For a point in the left boundary a point in the gap on the right boundary is searched, starting from the middlemost point to the extreme point of the *top* (*bottom*) part. As soon as a point is found it is joined and the gap structure is modified.

The joined point on the left boundary of right block is propagated to the right vertical edge of the right block following same way of *'routing inside a block'*. If the target is in the vertical river then the point is not joined with the gap of right block, instead it is stretched to the right block (See the figure 9).

If the right block does not have the gaps the paths are stretched to the upper (lower) horizontal river for the *top* (*bottom*) part (see the figure 9).



Edges from horizontal boundary to adjoining rivers

Edges from horizontal boundary inside the block

Edges from left vertical boundary inside the block

Edges from right vertical boundary inside the block

☆ Signifies aborted routing

✦ Signifies edges which were aborted in previous stage but routed in this stage

Fig. 8: Routing outside the boundary for horizontal edges

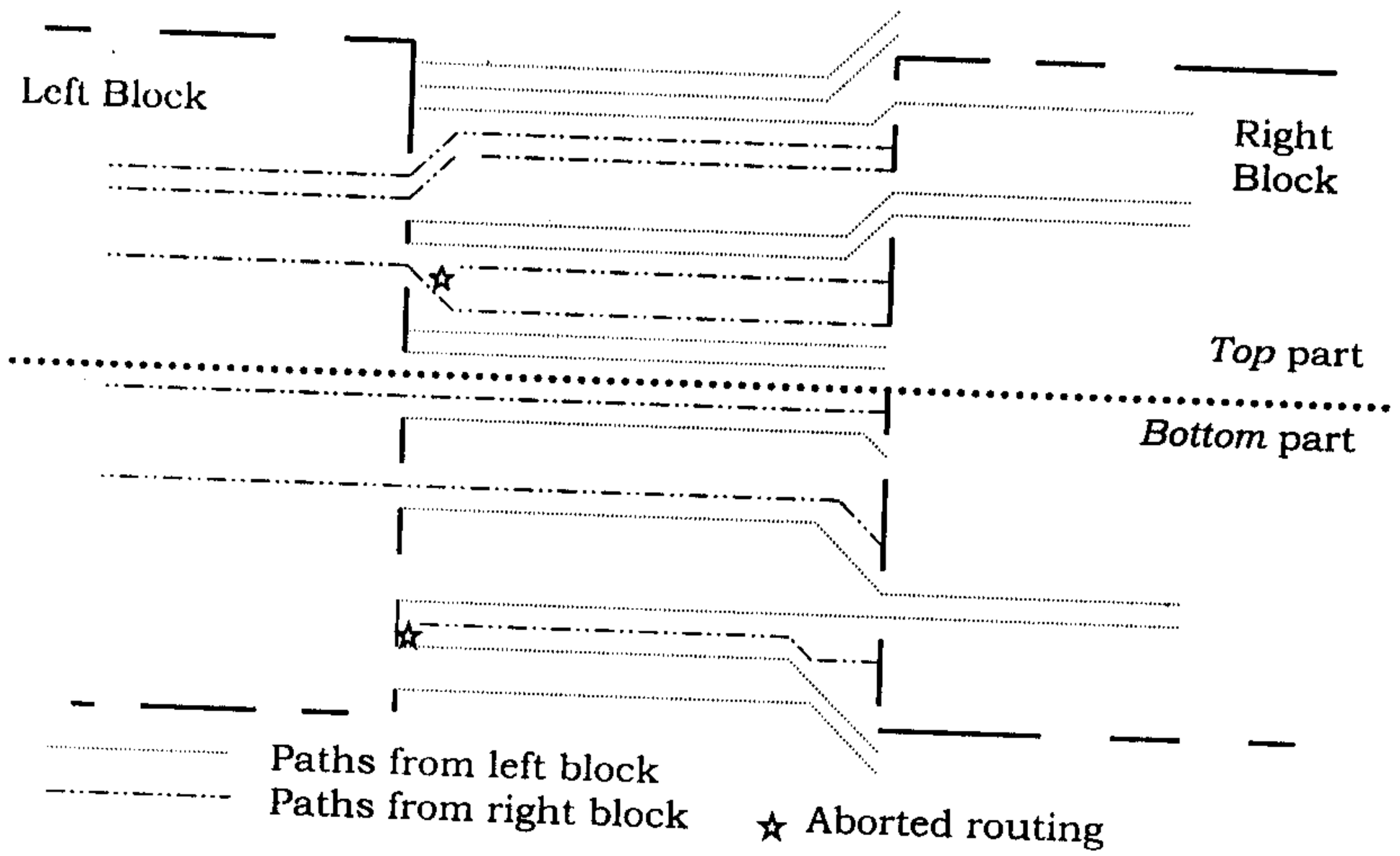


Fig. 9: Vertical boundary routing outside a block in phase1

All the points on the right vertical edge of the left block have been considered. Following the way all pairs of blocks are considered. Then for every block-pair the left vertical edge of right block is considered by the following way. The pairs are considered starting from the rightmost vertical river to the leftmost. In a vertical river, the pairs are considered starting from the topmost to the bottommost. Then the left vertical boundary of the right block is considered similarly with one exception. No edge is allowed to be stretched to the adjoining horizontal river. This is illustrated in Fig. 9.

Horizontal rivers

Now the horizontal rivers are considered in the left division of horizontal layer one after another starting from the topmost to the bottommost. A river is considered starting from rightmost block-pair to the leftmost block-pair with which it is adjoining.

For a block-pair the horizontal river in between is divided horizontally into two equal halves *top* and *bottom*. Upper block uses the *top* part and

lower block uses the *bottom* part. For each block-pair the lower (upper) block edges which have been put to the horizontal river previously, are stretched to the right boundary of the *left* division of the horizontal layer, if the target is in the right division of the horizontal layer. Otherwise it is stretched up to its target. All the edges may not be stretched to the boundary of the *left* division or to its target as the river capacity is finite. In this case the competition between two colliding net is resolved by the following rule. The net, which has its target nearer, is given the preference and hence it wins. Routing of other net is aborted. Fig. 10 illustrates the situation for top boundary of *left* division.

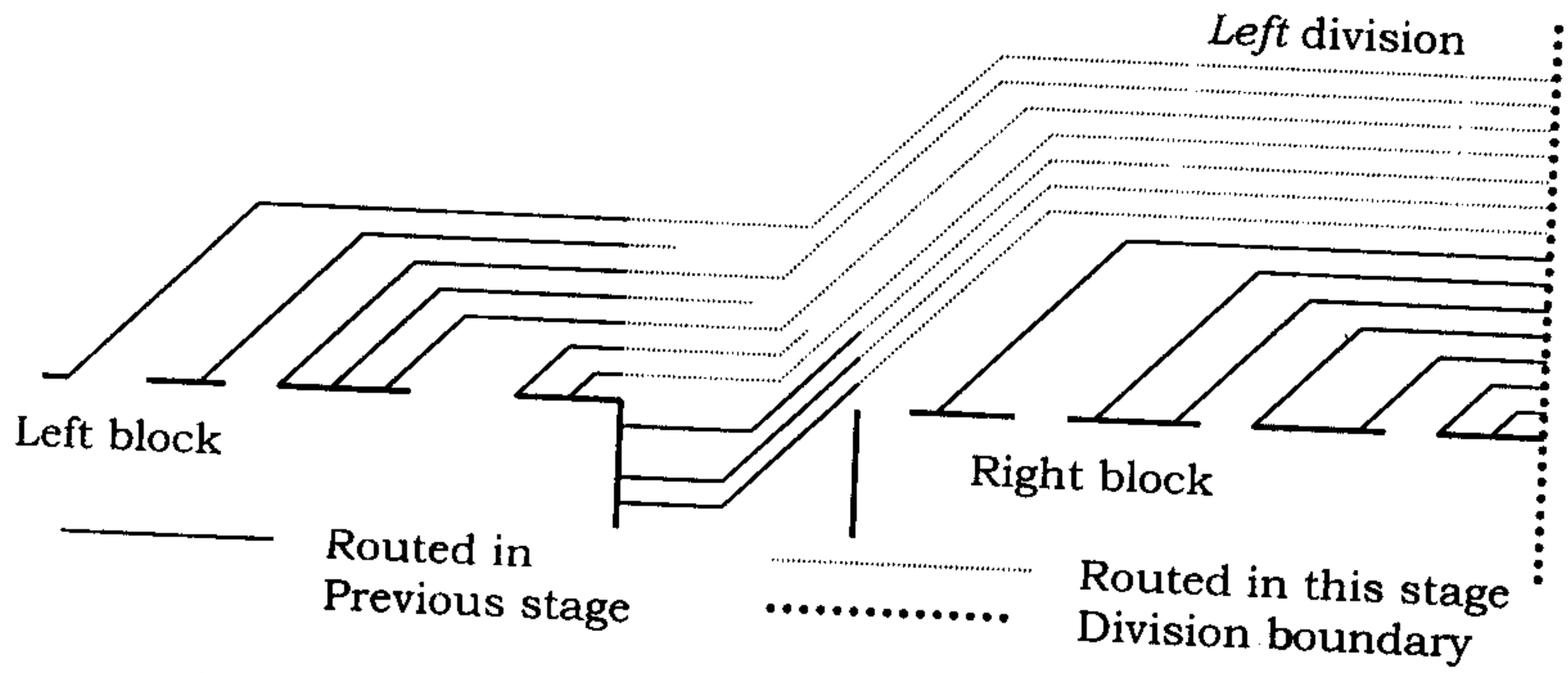


Fig. 10: Horizontal boundary routing outside of a block in phase1

After this step is complete the phase 1 of routing is over. Phase 1 maybe done simultaneously in the two divisions (left and right division of horizontal layer or left and right division of vertical river). But phase 1 routing is done first in horizontal layer. The horizontal edges may be shifted to the top or bottom. As a result the target of the vertical edges may be changed. So the targets of all the vertical edges of the nets, which are not aborted during the phase 1 routing of horizontal layer are updated. Then the phase 1 routing of vertical layer is done. The order of phase 1 routing in horizontal and vertical layer may be changed without any effect on the routing.

4.2.2 Phase 2

After phase 1 of routing the following situation arises. Inside routing of all the division i.e. *left* and *right* division horizontal layer and *top* and *bottom* division for vertical layer are over. In horizontal layer some edges of *left* division have target edges in the *right* division and vice-versa. Similarly in vertical layer some edges from *top* division have target in *bottom* division and voce-versa. Before starting phase 2 of routing targets of horizontal edges are updated, because just before it phase 1 routing of vertical edges has been done.

Phase 2 routing routes the edges, which starts in one division (say at top division of vertical layer) and have target edge in opposite division (i.e. bottom division of vertical layer). Here only the case of horizontal layer is illustrated. Phase 2 routing of vertical layer may be done similarly.

Block-pairs

Side by side blocks, which are separated by the vertical strip, are considered first starting from the topmost pair to the bottommost. For each block in a pair, how many edges have their target to the opposite division, are calculated. Here it is assumed that left block have more edges whose target are in the opposite (*right*) division. Other case may be handled similarly. How many blank points are available in the gaps on the left vertical side of the right block is calculated. If the number of blank terminals are more than the number of edges, which have target in the *right* division then the edges are distributed uniformly in the blank points of the gaps on the vertical side if right block. Edges that have its target in the strip are extended up to the right block.

If the available blank points are less in number of edges, which have target in the right division then extreme edges are pushed to the adjoining horizontal river and intermediate edges are connected to the blank points, if the horizontal river is not full. If the rivers are full, the routing of nets, which contain these extreme edges, is aborted.

Now the edges of the right block are considered, whose target is in the *left* division of horizontal layer. Such edges are searched from the left vertical side of right block starting from the topmost point to the bottommost point. As a edge is found, blank points from the gaps of right vertical side of the left block is searched such that it can be joined without colliding with any other edge. Topmost point from the available points is selected, joined and the gap structure is modified. If no poin5t is available then the routing of the net to which this edge belongs to is

aborted. If the target of the edge is in the strip then it is extended to the right vertical side of the left block.

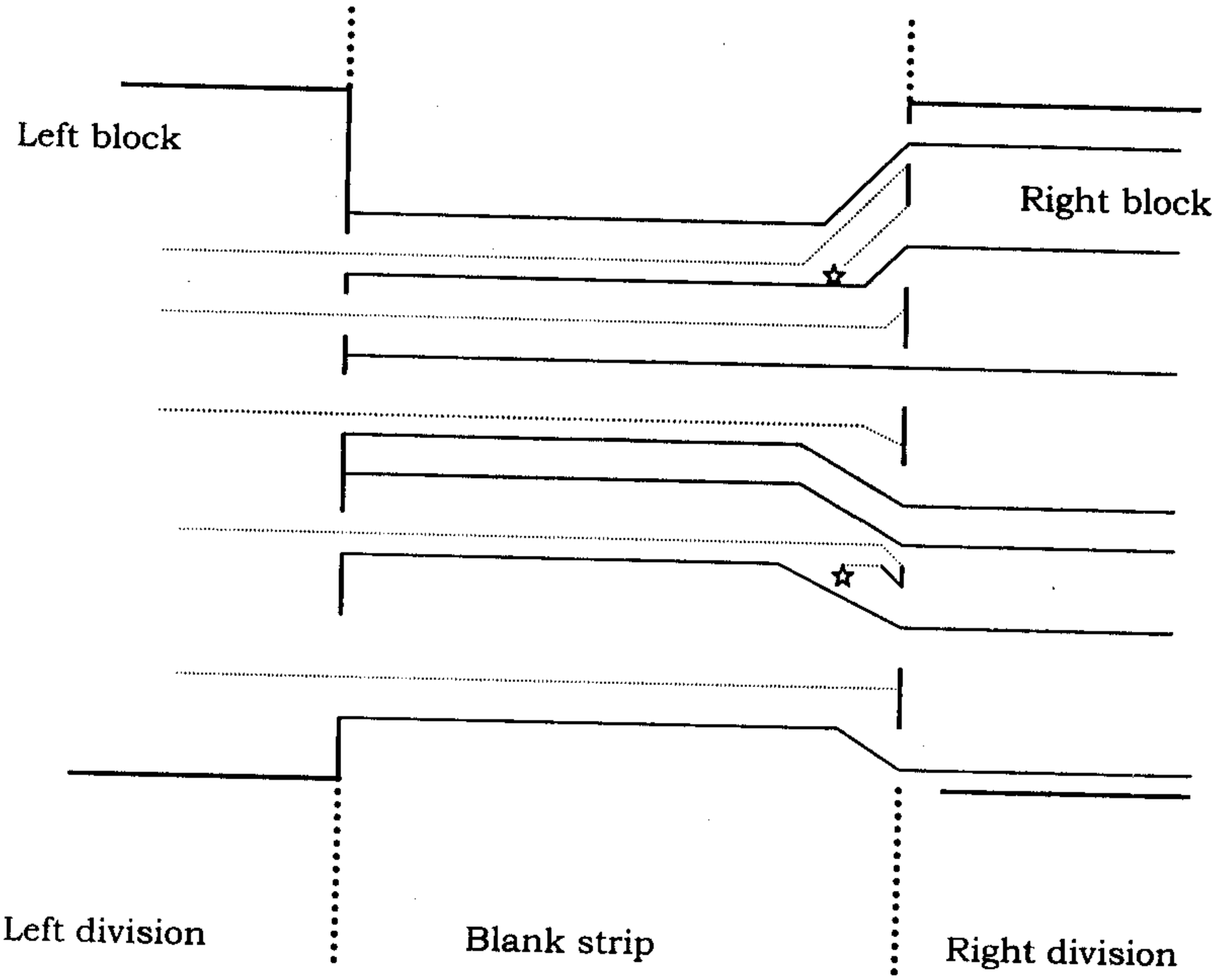


Fig. 11: Vertical boundary routing in phase2

Now the right vertical side of the left block has some edges, which have their target to the left and left vertical side of the right block has some edges, which have their target to the right. These edges may be routed following the same way of *'phase 1 routing'*. Some edges may not be routed to their target. Routing of net, which contain these edges, are aborted. This is illustrated in the Fig. 11.

Horizontal rivers

In most of the cases the rivers become full before starting this stage. In this discussion the rivers will be assumed as full. If they are not then the case may be handled easily.

Rivers are collided head-on; the competition is resolved by considering the remaining distance from their targets. The edge, which has the least distance to be routed, is given preference and it wins. Routing of the net, which contains the defeated, is aborted. The winner uses the edge of the defeated net.

After the phase2 routing of horizontal layer is completed, all the vertical edges of non-aborted edges update their targets. Then they start their phase 2 routing.

Chapter 5

Data structure

The edges are stored in an array of structures, which contains the following fields. A) Starting points. B) Destination edge. C) Net to which it belongs. D) Type of edge (horizontal or vertical). Updating of target is done in B field.

Gaps of each are kept in a doubly connected link list such that starting from any gap both way (clockwise or anticlockwise) can be traversed for searching a suitable gap. Each entry contains the following fields. A) Starting point of gap. B) Finishing point of gap.

Blocks are kept in an array of structures, which contains the following fields. A) Left bottom corner point. B) Right top corner points.

Substrate information is kept in a two dimensional array of substrate size. A grid point is represented by the array member of indices equal to the coordinate of the grid point. The member keeps the number of net, which uses the grid point. If the grid point is blank, a negative number is stored.

Chapter 6

Experimental results

This heuristic algorithm was tested on a standard benchmark example. In Table 1 we give the short description of the benchmark.

Table 2 and table 3 compares the performance of this algorithm with the results obtained by VAR[3] and SEGRA[1].

Table 4 gives the CPU time taken by our algorithm to run.

Examples	# Chips	# nets	# pins	Grid size
Test 1	4	500	1000	300 X 300

Table1: A benchmark example.

Example	Number of vias			
	Alg. of Das, Nandy and Bhattacharya [5]	VAR	SEGRA	OUR method
Test1	1520	2250	2080	1290

Table2: Comparison of number of vias

Example	Total wire length			
	Alg. of Das, Nandy and Bhattacharya[5]	VAR	SEGRA	Our method
Test1	102,238	104,128	103,682	106,752

Table 3: Comparison of wire length

CPU time taken by our Algorithm
1.319405 e^{-301} seconds

Table 4: CPU time

Reference

[1] Y – J Cha, C. S. Rim and K. Nakajima, " A simple and efficient greedy multi – layer router for MCMs" *Proc. Int'l Symp. On physical Design.* Pp 67-72. 1997.

[2] K. Y. Khoo And J. Cong. " A fast multiplayer general area router for MCM design," IEEE trans. *On circuit and systems II,* pp 841 – 851, Nov. 1992

[3] K. Y. Khoo and J. Cong, "An efficient multiplayer MCM router based on four-via routing", *IEEE TCAD,* vol. 14, pp 1277 – 1290, Oct. 1995

[4]N. Sherwani, S. Bhingarde and A. Panyam, *Routing in the third dimension:* From VLSI chips to MCMs, IEEE press, Piscataway, NJ, 1995.

[5] S. Das, S. C. Nandy, B. B. Bhattacharya, "High performance MCM routing: A new approach", *12 th International Conference on VLSI Design* – January 1999, pp 564 - 569.

[6] N. A. Sherwani, *Algorithm for VLSI physical design Automation,* Kluwar Academic Publishers.