TR-87-013

A GRIDLESS PATTERN POUTER WITH GLOBAL VIA ASSIGNMENT

TR-87-013

A Gridless Pattern Router with Global Via Assignment

Y. S. Kuo

Institute of Information Science

Academia Sinica, R. O. C.

C. J. Ho, Chris Pu and Y. P. Sun

The Intertek Corporation, R. O. C.

Address of Correspondence

Dr. Y. S. Kuo

Institute of Information Science

Academia Sinica

Nankang, Taipei, Taiwan

Republic of China

A Gridless Pattern Router with Global Via Assignment

5. PCB Placement and Routing

Abstract

A gridless pattern router is described. The router can generate connecting paths having no more than two vias. Wires on each layer can have turns and 45^{o} segments are allowed. A global via assignment procedure is implemented to minimize the via usage. The concept of degree of freedom is introduced for the ordering of pin pairs.

1. Introduction

Pattern routing has been recognized to be one of the fundamental routing schemes for the printed circuit board (PCB) layout. [9] [1] [6]. Given a pair of pins to be connected, a pattern router tests prescribed patterns or shapes in a certain order until it obtains a route connecting them. Such a router has several advantages. It can be incorporated with an interactive router nicely for it is an auto-router which mimics human layout designers very closely. A pattern router almost always generates good routes (eg. few through holes, short connections, etc.). Thus, in case that it can not complete all connections on a board, the human layout designer can complete the rest of the connections easily without making excessive modification to the wires already completed by the pattern router.

Most commercially available auto-routers rely on the concept of routing grids. Their internal data structure, or model of the PCB is a cell map exemplified by the well-known maze router [5]. However, the recent advance in fine-line technology has set a limit to this approach [3]. The layout of fine-line PCBs requires fine grids and thus a large cell map. As the traces of PCBs become finer and finer, gridded routers could expend an enormous amount of storage and their run times might be prohibitive.

To cope with the increasing amount of data associated with fine-line PCBs, we adopt the gridless routing approach. All objects on a PCB are modelled as polygons which are represented by the coordinates of their vertices. Round pins or vias are approximated by octagons that circumscribe them. Using this data representation, we can obtain an accuracy to the order of 10^{-2} mil on a 32-bit work station. One commercial PCB router has been reported which uses this gridless approach [2]. But we are not aware of any gridless pattern router.

The performance of a pattern router depends largely on how many different wiring patterns it considers. If the router considers many wiring patterns, its connection completion rate can be high but it will consume much computation time. On the other hand, if the router considers few wiring patterns, its connection completion rate becomes low but it runs fast. To guarantee generating good routes with minimal vias, our pattern router only considers wiring patterns that have no more than two vias. But wires on each layer can zigzag (i.e. have turns) and 45^o segments are allowed to improve the system's routability. Using this strategy, we hope to build a router which has fairly good completion rate but still runs efficiently. Plane-sweep [8] and branch-and-bound [7] techniques have been used for this goal.

A pattern router usually makes connections sequentially. A connection path made earlier may block some later connections. To alleviate this ordering problem, we have a via assignment procedure which can determine the locations of vias (in other words, the outlines of paths) globally. The PCB is first divided into a number of regions. Then vias are assigned to regions using a maximum matching algorithm [7] in order that each region will have roughly the same number of pins and vias. This via assignment procedure is essentially a global router

[10]. It also has the effect of making one-via connections for as many unaligned (diagonal) pin pairs as possible.

Another feature peculiar to this router is the concept of degree of freedom (dof). The dof of a pin is a measure of its routability in its locality. When a pin's dof drops to 0, the pin can hardly be connected to any other pin. For a pair of pins to be connected, the dof is defined in terms of the dofs of the two pins. Thus the dof of a pin pair is an indication of the freedom (number of possible ways) that the two pins can be connected. As pin pairs are connected one after another, the dofs can be updated dynamically, and the pin pair to be connected next is the one which has the least dof. This ordering scheme has the advantage that pin pairs tend to be routed before they become unroutable.

In the next section, we introduce the overall structure of the pattern router. Then in Section 3, we consider the algorithms for finding appropriate connecting paths. The via assignment and ordering schemes are decribed in Section 4. Finally, we shall demonstrate some test results in the last section.

2. The Pattern Router

The pattern router considers 4 different wiring patterns:

(1) no-via paths for aligned pin pairs

(2) one-via paths for unaligned pin pairs

(3) two-via paths inside the smallest enclosing rectangles

(4) two-via paths outside the smallest enclosing rectangles

These patterns are illustrated in Fig. 1, 2, 3 and 4, respectively. Note that the wires shown in these figures are only symbolic. The actual wires can zigzag and 45° segments are allowed.

Each layer of the PCB has a preferred direction and a nonpreferred direction. If its preferred direction is horizontal, then its nonpreferred direction is vertical, and vice versa. A physical path on one layer can proceed indefinitely in the layer's preferred direction, but it must fall inside a banded zone limited in the layer's nonpreferred direction.

The wiring patterns consist of horizontal and vertical wires. Each horizontal (vertical) wire is realized by a physical path on a layer whose preferred direction is horizontal (vertical).

The pattern router first partitions nets into a set of pin pairs using a minimum spanning tree algorithm [7]. These pin pairs are then connected with the 4 wiring patterns.

Algorithm PatternRouter:

(1) Find no-via paths for pin pairs which are aligned horizontally or vertically. A pin pair is aligned if the two pins' ordinates or abscissae differ by no more than a fixed small value. The pin pairs which fail to be connected at this step will be reconsidered at step 4.

(2) Find one-via paths for unaligned pin pairs. Each pin pair has two possible locations for the single via. Run the global via assignment procedure to determine the via locations before finding the connecting paths. The pin pairs which fail to be connected at this step will be reconsidered at step 3.

(3) Find two-via paths for the remaining unaligned pin pairs. Each path is inside the smallest rectangle enclosing the two pins.

(4) Find two-via paths for all the remaining pin pairs. For each pin pair, the two vias are placed outside the smallest rectangle enclosing the two pins.

The main data structure used by this router is a 4-dimensional binary tree [4][2]. All objects such as pins, vias and wire segments, etc. are stored as rectangles in the leaves of the tree. Such a tree structure facilitates the efficient retrieval of objects which fall inside a rectangular query window.

3. Searching for Paths

Even though there are 4 different wiring patterns, the actual connecting paths are found via one common structure, the access graph. Consider a rectangular region within which there are some polygonal obstacles as shown in Fig. 5(a). The free space, i.e. the space outside the obstacles, can be partitioned into a number of trapezoids called the free blocks. These free blocks are the nodes of the access graph. Two free blocks are adjacent (connected by an edge) if they can communicate to each other directly, i.e. they have sides overlap. The partitioning of the free space in Fig. 5(a) is shown in Fig. 5(b) and the associated access graph is shown in Fig. 6. Here the rectangular region is on a layer whose perferred direction is horizontal. Thus the trapezoids have their parallel sides vertical.

The access graph can be constructed by using the typical plane sweep algorithm in computational geometry [8]. Assume that the region is on a layer whose preferred direction is horizontal. The vertices of the obstacles are first sorted by their abscissae. Then the trapezoids can be formed one after another as a vertical line sweeps from left to right halting at each vertex. The sides of the obstacles that intersect with the sweep-line can be kept in a balanced binary tree and updated dynamically. With this data structure, this algorithm runs in time O(n log n) where n is the number of vertices of the obstacles.

Given a rectangular region where two pins to be connected are positioned at the two opposite vertical sides of the rectangle (Assume that the region is on a layer whose preferred direction is horizontal), the path-finding problem is to find a connecting path which has the minimal total wire length in the vertical (i.e. nonpreferred) direction. To be formal, the length of a wire connecting points (x_1,y_1) and (x_2,y_2) is defined to be $|y_1-y_2|$, and the path length is the sum of the lengths of the wires on the path. We assume that a path can not have a backward turn in the horizontal (i.e. preferred) direction.

The two pins to be connected are located at two free blocks, say S and T. Then a physical path for the pin pair determines a chain of free blocks in the access graph connecting S and T. Conversely, given a chain of free blocks connecting S and T (no backward turn), we can construct a physical path passing through these free blocks by the following greedy method.

Algorithm FindPath1

Generate a physical path from left to right, starting from the leftmost free block. As shown in Fig. 7(a), the physical path enters the left free block at point p. If a horizontal line passing through p can enter the right free block from its left vertical side, then the physical path proceeds horizontally as shown in Fig. 7(b). Otherwise, the physical path detours and enters the right free block from one of its vertices as shown in Fig. 7(c). The vertex selected is the one that causes a smaller detour.

Theorem 1. Given a chain of free blocks connecting S and T , Algorithm

FindPath1 can generate a shortest physical path among all the paths which pass through these free blocks.

Algorithm FindPath1 is oversimplified. First, the design rules such as wire width and clearance must be taken into account. Second, the access graph can have more than one chain of free blocks connecting S and T. When the graph has many chains of free blocks, determining a shortest physical path does not seem to be that simple. (Determining a shortest chain of free blocks is meaningless since the length of a chain is not quite related to the length of a physical path.)

In general, we use a branch-and-bound algorithm for finding a shortest physical path. In this algorithm, the access graph is traversed in a depth-first manner, and different chains of free blocks are exploited. During the traversal, the physical paths passing through these chains of free blocks are computed and compared. Eventually the shortest physical path is picked up among all of these physical paths.

```
Algorithm FindPath (free block B, point p(x,y), bound b)
/* FindPath is recursive.  p(x,y) is a point positioned at the left   */
/* side of free block B.  FindPath can generate a shortest physical   */
/* path P leading from p to the target pin t($x_t$,$y_t$) with path length */
/* |P| < b if such a path exists.                                    */
If the target pin t is positioned at the right side of B
then return P as a path connecting p and t inside B
else let $B_i$, i= 1,2,...,m, be the free blocks adjacent to B and to its
     right as shown in Fig. 8.
```

```
        For each i, compute the path leading from p and entering $B_i$
        as in Algorithm FindPath1.  Let $p_i(x_i, y_i)$ be the point at which the
        path enters $B_i$.
        Order the free blocks $B_i$ by their estimated path lengths
            $d_i = |y_t - y_i| + |y_i - y|$
        Assume that $B_i$'s have been reassigned subscripts such that
            $d_1 \le d_2 \le \dots \le d_m$
        For each i, i=1,2,...,m
            if $d_i < b$ then
                call Algorithm FindPath ($B_i$, $p_i$, $b - |y_i - y|$ )
                if FindPath returns a path $P_i$ then
                    $P = (p, p_i) \| P_i$            /* concatenate */
                    $b = |P|$
                endif
            endif
        endfor
    endif
```

Among all the paths passing through B_i, d_i is a lower bound to the path lengths. Thus, if $d_i \geq b$, there is no need to consider B_i any more. This cuts off a large portion of the search tree. On the other hand, since we rearrange B_i's in ascending order of d_i, short paths can be explored as early as possible.

Theorem 2. Algorithm FindPath can generate a shortest physical path for a pin pair.

For an aligned pin pair, Algorithm FindPath can generate a no-via path directly. To create a one-via path for an unaligned pin pair, we first construct the access graphs for two intersecting rectangular regions as shown in Fig. 9. Then inside the intersection area, a point is located for the via. The point selected must be inside two intersecting free blocks of the access graphs. After determining the via location, Algorithm FindPath generates the physical paths for the two pin/via pairs.

The generation of two-via paths can be accomplished through the combination of access graphs and Algorithm FindPath as well. But we will not consider the details here.

4. Via Assignment and Ordering of Pin Pairs

After the pattern router has completed the no-via paths for aligned pin pairs, it intends to make one-via connections for as many unaligned pin pairs as possible. This has the effect of minimizing the via usage and is accomplished by the global via assignment procedure.

The via assignment procedure distributes the vias uniformly over the entire PCB in order that as many one-via paths can be accommodated on the board. The board is divided into a number of equal-area regions as shown in Fig. 10. For each unaligned pin pair, there are two possible locations for the single via, each of which must fall inside a region. Thus we can construct a bipartite graph $G = (U,V,E)$ as follows: The nodes of U correspond to the pin pairs and the nodes of V to the regions. Each node of U is adjacent to two nodes of V representing the two possible regions to be assigned to a pin pair for its via. The bipartite graph associated with the PCB in Fig. 10 is shown in Fig. 11.

Associated with each region (in other words, each node of V), there is a region pin density defined as the sum of the wire counts of the pins inside the region, where the wire count of a pin is the number of pin pairs to which the pin belongs. If certain vias have been assigned to a region, then the region's total density becomes the region pin density + 2 * number of vias since the wire count of a via is 2. Suppose that a density bound DB is imposed on all regions. Then the number of vias that can be assigned to a region is at most
$\lfloor(\text{DB} - \text{region pin density})/2\rfloor$.

Given a density bound DB, the via assignment problem is to find an edge set $M \subseteq E$ of maximal cardinality such that

(1) Each node of U is incident to at most one edge of M

(2) For each node v of V, the number of edges of M incident to v is bounded by $\lfloor$(DB - region pin density)/2$\rfloor$.

In practice, we work with the via assignment problem first with a small density bound. Then the density bound is raised gradually to increase $|M|$ until eventually $|M| = |U|$.

The via assignment problem can be treated as a special matching problem, even though it is more general than the ordinary maximum matching on a bipartite graph. (Replacing the bound $\lfloor$(DB - region pin density)/2$\rfloor$ by 1 in (2), the via assignment becomes the maximum matching.) In fact, like the maximum matching, the via assignment problem is a special case of the network maximum flow. Thus the via assignment can be solved as maximum matching and maximum flow by the method of augmenting path.

On the bipartite graph $G = (U,V,E)$, let M be an edge set satisfying (1) and (2). A node $u \in U$ is exposed if u is not incident to any edge of M. A node $v \in V$ is exposed if v is not incident to $\lfloor$(DB - region pin density)/2$\rfloor$ edges of M. Nodes which are not exposed are called closed. A path $[u_1,v_1,u_2,v_2,\ldots,u_k,v_k]$ is called augmenting if the edges $(u_i,v_i) \in E-M$, $i = 1,2,\ldots,k$, and $(v_j,u_{j+1}) \in M$, $j = 1,2,\ldots,k-1$, and all nodes on the path are closed except that u_1 and v_k are exposed. Algorithms for finding augmenting paths can be found from the reference [7]. Thus we shall not deliberate here.

The pattern router makes connections sequentially. Pin pairs to be connected are ordered according to their degrees of freedom (dof). Each pin has a h-dof (horizontal degree of freedom) and a v-dof (vertical degree of freedom). The h-dof is the number of free tracks right above and right below the pin. Thus the h-dof of a pin is at most 2. When a wire passes right above the pin, the pin's h-dof drops to 1. The v-dof is defined similarly in the vertical direction. The dof of a pin is the sum of its h-dof and its v-dof. When a pin's dof drops to 0, the pin can hardly be connected to any other pin.

Given a wiring pattern for a pin pair, the dof of the pin pair is defined in terms of the dof's of the two pins. For an horizontally aligned pin pair, the dof of the pin pair is the product of the h-dofs of the two pins. For an unaligned pin pair to be routed as shown in Fig. 12, the dof of the pin pair is h-dof(p_1) * v-dof(p_2). Similarly, the dofs can be defined for other wiring patterns. Thus the dof of a pin pair is an indication of the freedom (number of possible ways) that the two pins can be connected.

As pin pairs are connected one after another, the dofs can be updated dynamically, and the pin pair to be processed next is the one which has the least dof. This ordering scheme has the advantage that pin pairs tend to be routed before they become unroutable.

5. Test Results

The pattern router described in this paper is currently being developed at the Intertek Corporation. This prototype system is programmed in PASCAL and runs on a 68000-based HP-3000. Even though this system is not fully completed yet (e.g. The generation of 2-via paths is still under development), the outcome of the partial system has been encouraging.

The routing of a small 2-layer digital board is shown in Fig. 13. The main characteristics are:

Board size 6.7 in x 3.9 in

component density 0.62 in^2/equivalent 14 pin D.I.L.

wire width/clearance 12 mil

339 pin pairs

completion rate 88% (only no-via and one-via paths)

run time 77 minutes on a 68000-based HP-3000.

References

1. T. Asano, "Parametric Pattern Router", Proc. 19th Design Automation Conference, 1982.
2. A. C. Finch, et al., "A Mothod for Gridless Routing of Printed Circuit Boards", Proc. 22nd Design Automation Conference, 1985.
3. A. Finch and G. Symonds, "Problems that Advanced Technologies Bring to Routing", Proc. Automated Design and Engineering for Electronics West, March 1986.
4. U. Lauther, "A Data Structure for Gridless Routing", Proc. 17th Design Automation Conference, 1980.
5. C. Lee, "An Algorithm for Path Connections and its Applications", IRE Trans. on Electronic Computers, Sept. 1961.
6. H. Mori, et al., "Advanced Interactive Layout Design System for Printed Wiring Boards", in Hardware and Software Concepts in VLSI (ed. G. Rabbat), Van Nostrand Reihold Co., 1983.
7. C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization, Prentice-Hall, Inc., 1982.
8. F. P. Preparata and M. I. Shamos, Computational Geometry, Springer-Verlag New York Inc., 1985.
9. J. Soukup and J. Fournier, "Pattern Router", Proc. International Symposium on Circuits and Systems, 1979.
10. J. Soukup, "Circuit Layout", Proc. of the IEEE, Oct. 1981.

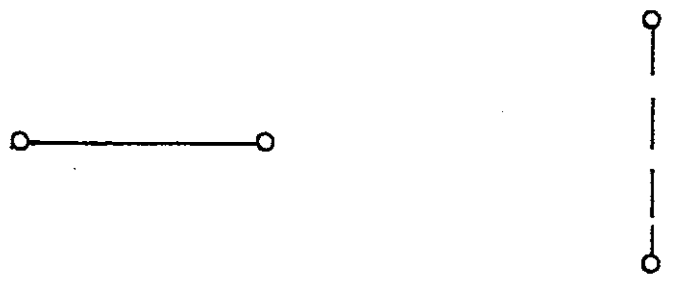


Fig. 1. No-via path

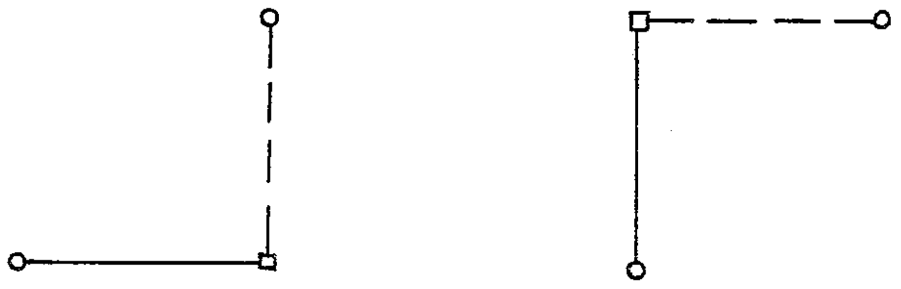


Fig. 2. One-via path

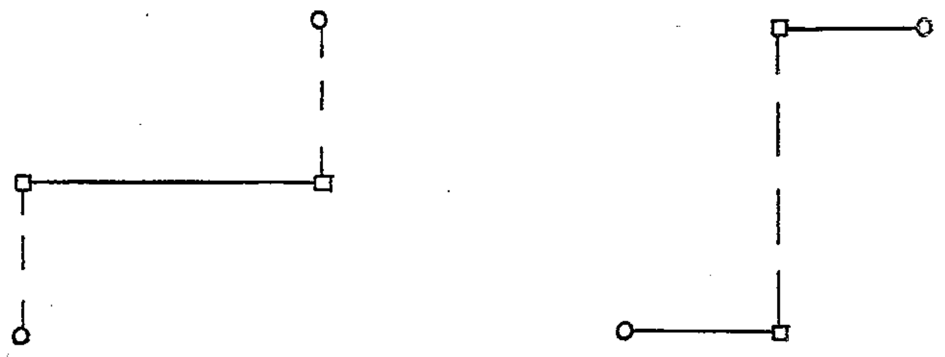


Fig. 3. Two-via path
inside enclosing rectangle

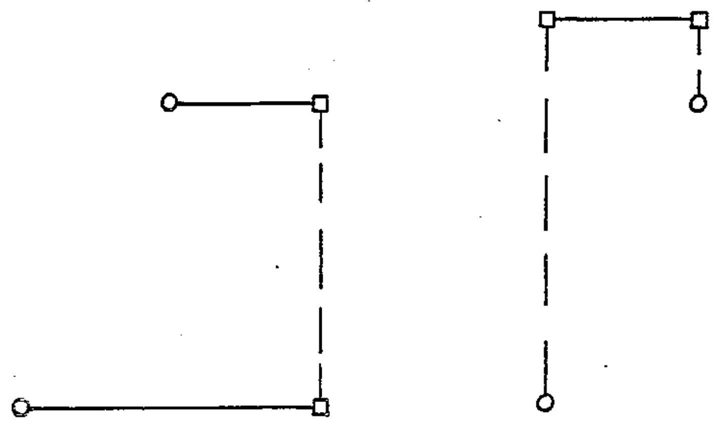
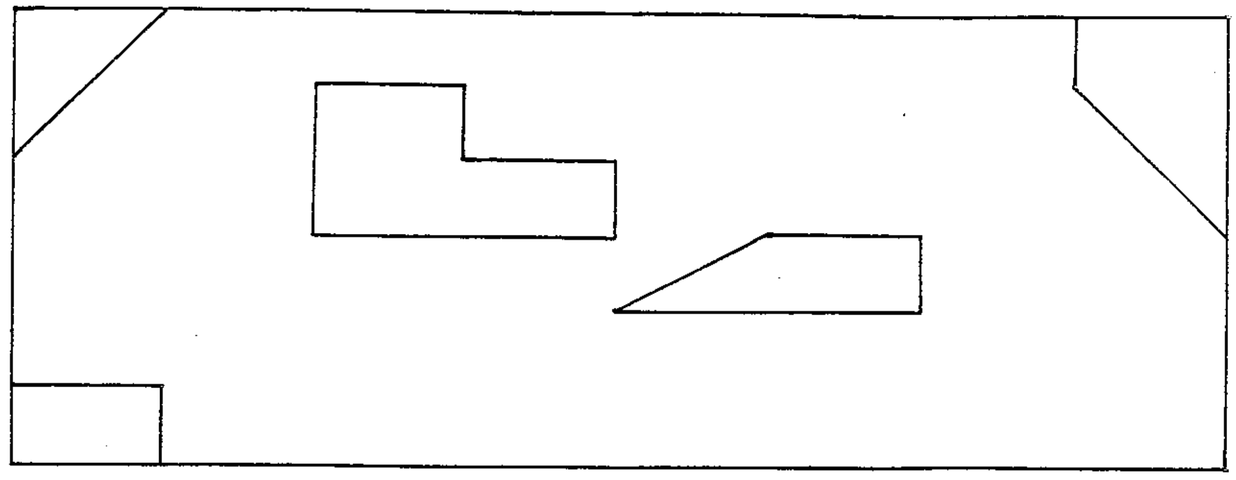
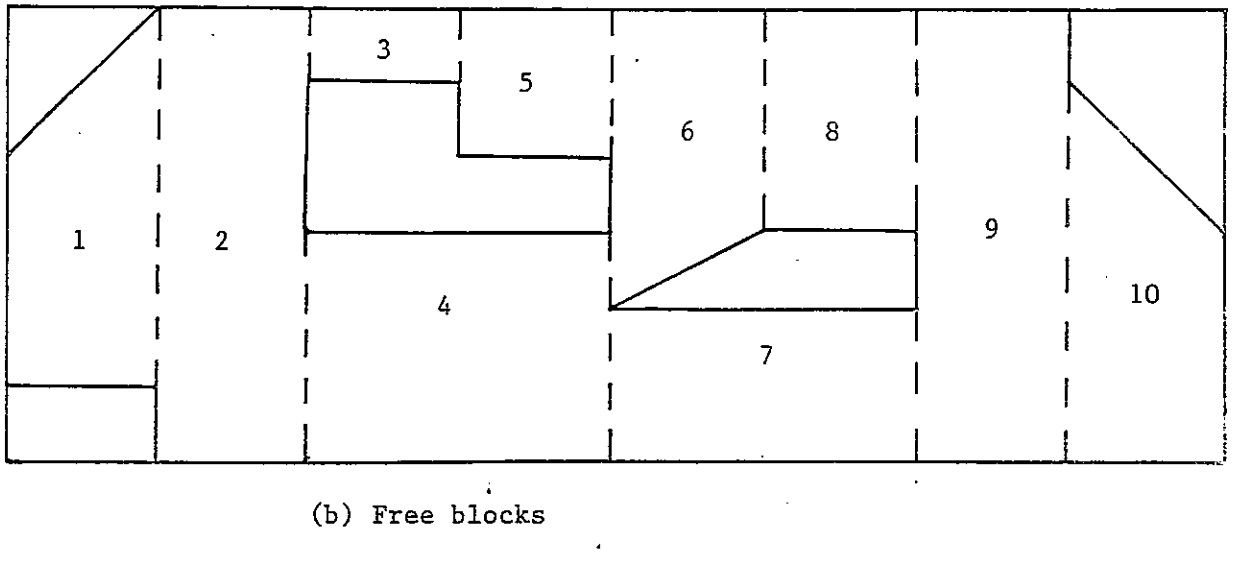


Fig. 4. Two-via path
outside enclosing rectangle



(a) Obstacles



(b) Free blocks

Fig. 5. Partitioning of free space

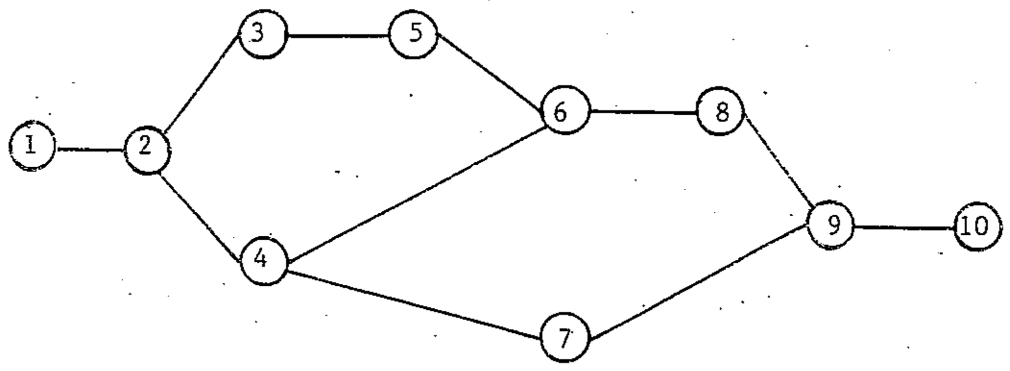


Fig. 6. Access graph

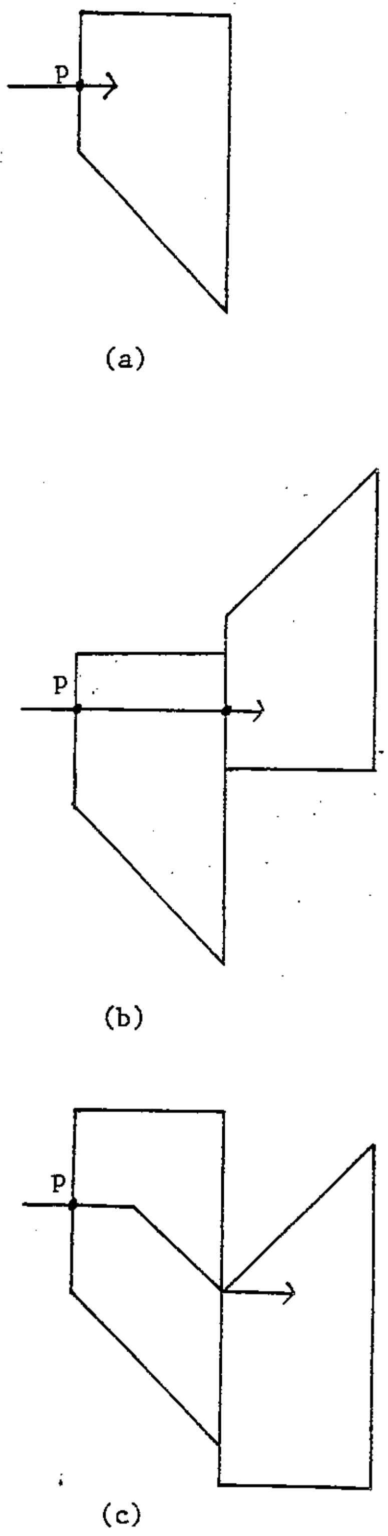


Fig. 7. Illustration for FindPath1

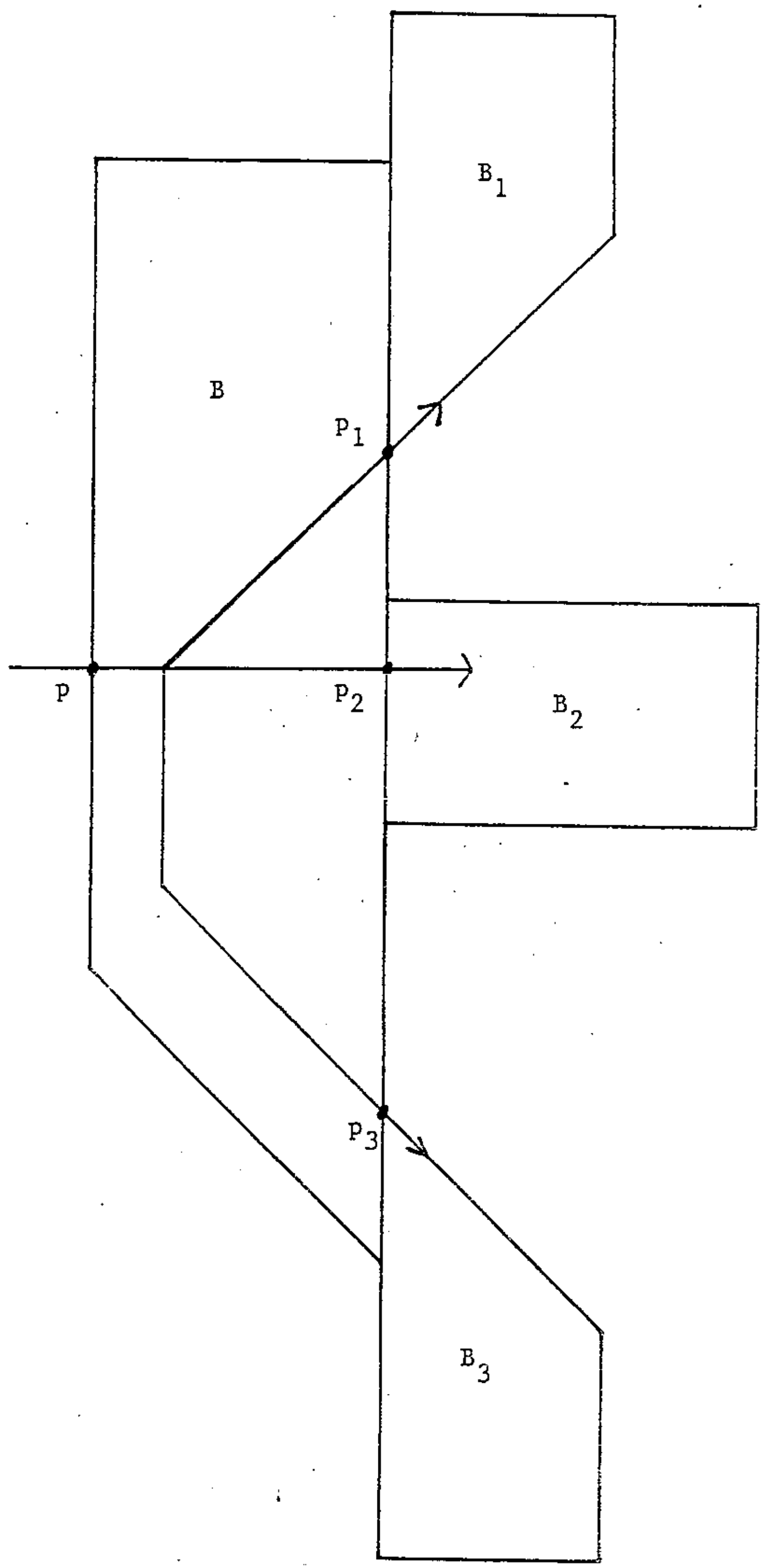


Fig. 8. Illustration for FindPath

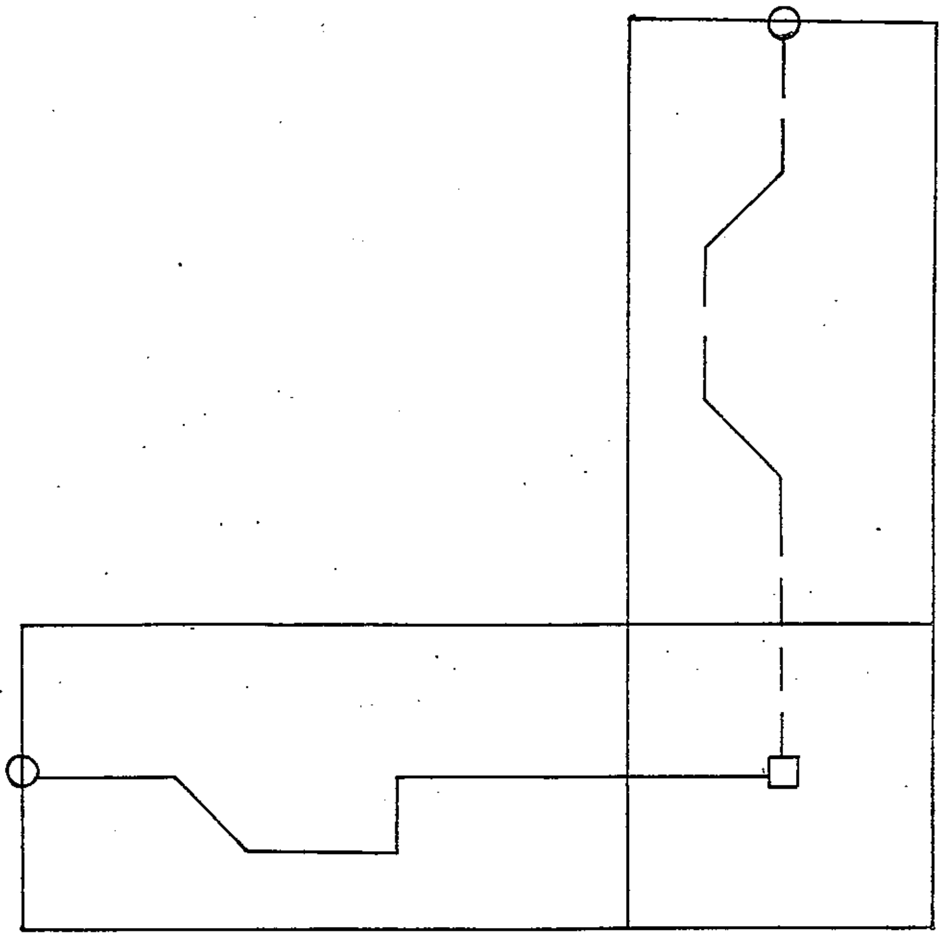


Fig. 9. Generation of one-via path

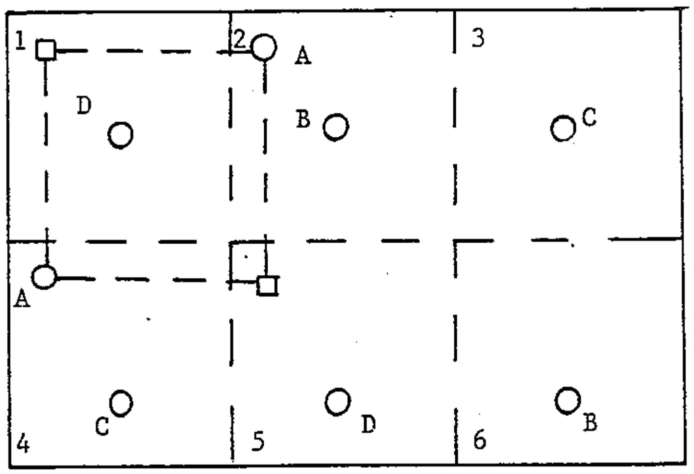


Fig. 10. Pin pairs on a board

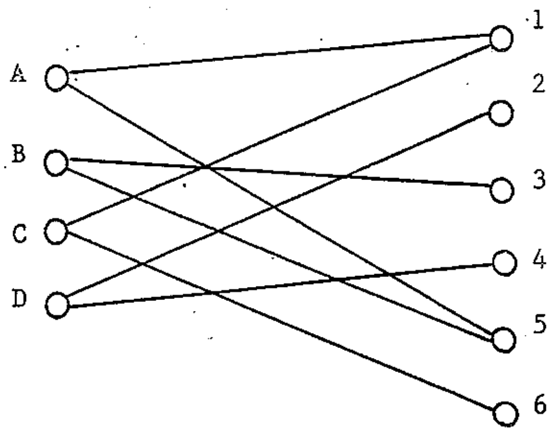
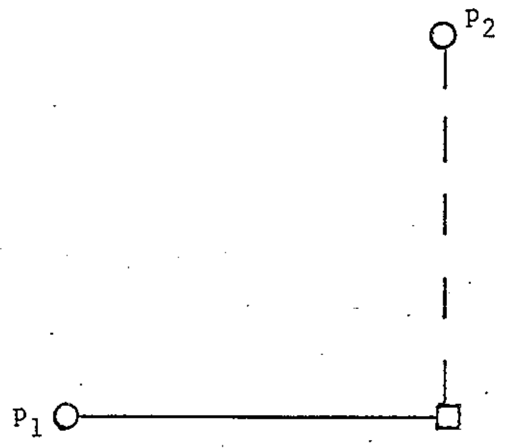
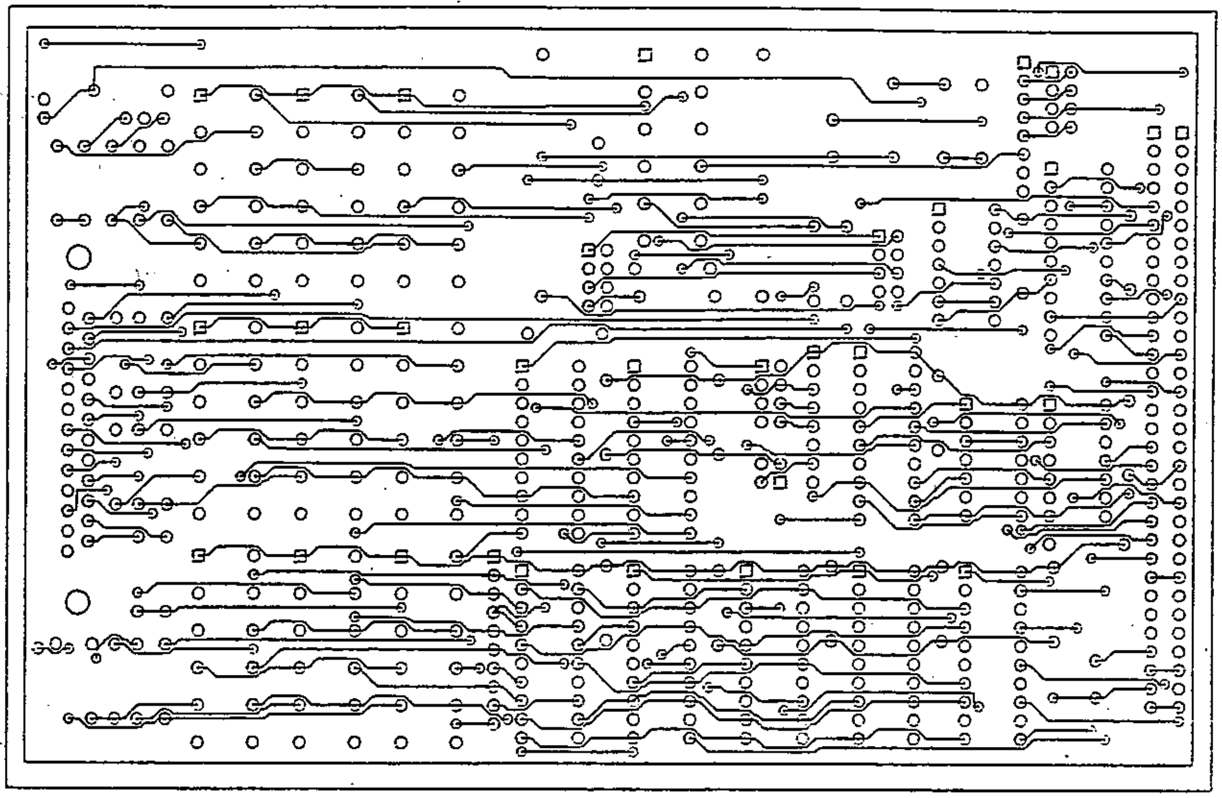


Fig. 11. Bipartite graph



dof(pin pair) = h-dof(p_1) * v-dof(p_2)

Fig. 12. Dof of a pin pair



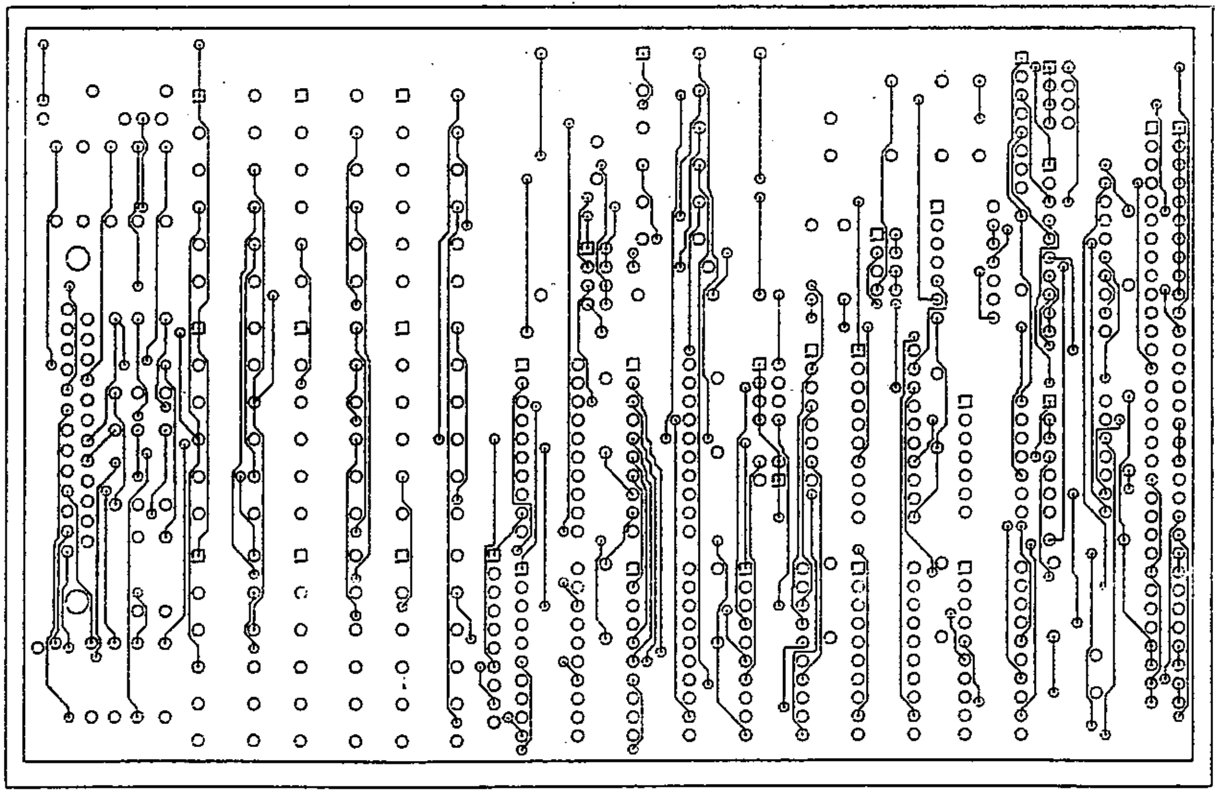


Fig. 13. Routing of a PCB