

Single-Row Routing in Narrow Streets

SANYONG HAN AND SARTAJ SAHNI, MEMBER, IEEE

Abstract—We develop fast linear time algorithms for single row routing when the upper and lower street capacities are less than or equal to three. A similarly fast algorithm is developed for the case when one of the streets has a capacity 1 and the other has an arbitrary capacity. Experimental results show that our algorithms are many times faster than previously developed algorithms.

I. INTRODUCTION

SO has proposed a systematic approach to the interconnection problem of large multilayer printed circuit boards in which pins and feedthroughs are uniformly spaced on a rectangular grid [6]. This approach consists of a systematic decomposition of the general multilayer wiring problem into a number of independent single layer, single-row routing problems. There are five phases in this decomposition [8], [5]:

1. Via assignment,
2. Linear placement of via columns,
3. Layering,
4. Single row routing,
5. Via elimination.

In this paper, we are concerned only with the fourth phase: Single row routing. In the single-row routing problem, we are given a set $V = \{1, 2, \dots, n\}$ of n nodes that are evenly spaced along a straight line; and a set $L = \{N_1, N_2, \dots, N_m\}$ of nets. Each net represents a set of nodes that are to be made electrically equivalent. The nets satisfy the following conditions:

- (i) $N_i \cap N_j = \emptyset, \quad i \neq j$
- (ii) $\bigcup_{i=1}^m N_i = \{1, 2, \dots, n\}$

jeN_i is a *touch point* of net i . The nets are to be realized in a single layer by the use of nonoverlapping wires that are composed solely of horizontal and vertical segments. Fig. 1(a) shows some of the legal ways to realize the net $\{1, 3, 6, 9\}$. The wire layout must satisfy the additional requirement that a vertical cut made at any point along the axis formed by the nodes can intersect at most one horizontal segment from each net. Thus the wiring of Fig. 1(d) is illegal.

The area above the line of nodes is called the *upper street* while that below this line is the *lower street*. Each street has *tracks* that run parallel to the line of nodes (Fig. 2). Horizontal wire segments must be layed in tracks and no track may hold

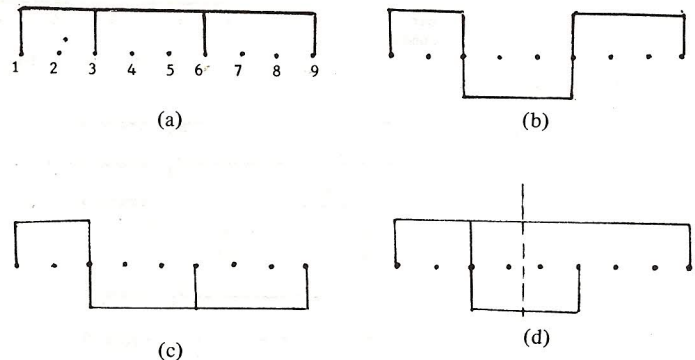


Fig. 1.

more than one wire segment at any point (of course, several non overlapping wire segments may be packed into the same track). Let K_u and K_l , respectively, denote the number of tracks in the upper and lower streets. Fig. 2 shows one way to realize the nets $(N_1, N_2, \dots, N_5) = (\{1, 7\}, \{2, 8\}, \{3, 6\}, \{4, 9\}, \{5, 10\})$ when $K_u = 2$ and $K_l = 3$.

In this paper, we are specifically concerned with obtaining net realizations (when they exist) given V (node set), L (net set), K_u (upper street capacity), and K_l (lower street capacity).

This problem has been studied before. Kuh *et al.* [3] and Tsukiyama *et al.* [9] developed necessary and sufficient conditions for a net set to be realizable. In [3], a simple construction is used to show that when K_u and K_l are sufficiently large, L is realizable. This construction, however, results in an algorithm of complexity $O(m!n)$ where $|V| = n$ and $|L| = m$. Raghavan and Sahni [4] have developed an algorithm of complexity $O(k! * k * n * \log k)$ where $k = K_u + K_l$ for this problem. This algorithm is quite practical when k is small but impractical when k is large. For the case $K_u \leq 2$ and $K_l \leq 2$, Tsukiyama *et al.* [9] have developed an $O(mn)$ algorithm. This algorithm is, however, slower than that of [4].

Here, we shall develop fast $O(n)$ algorithms for the case $K_u \leq 3$ and $K_l \leq 3$ as well as the case $K_l = 1$ and K_u arbitrary. Experimental results show these algorithms to be several times faster than that of [4] for these street capacities. The case K_u and $K_l \leq 3$ is actually solved by developing algorithms for the subcases $K_u = K_l = 2$; $K_u = K_l = 3$; and $K_u = 3, K_l = 2$. These algorithms together with that for $K_l = 1$ and K_u arbitrary cover all the possibilities for $K_u \leq 3$ and $K_l \leq 3$. Note that the case of $K_l = 0$ and K_u arbitrary is trivially solved in linear time.

Before presenting the algorithms, we introduce some notation. Following [4], nodes are classified by type:

- (a) Node i is of *type B* if it is the left extreme node of a net,

Manuscript received April 22, 1983. This work was supported in part by the National Science Foundation under Grant MCS 80-005856.

The authors are with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

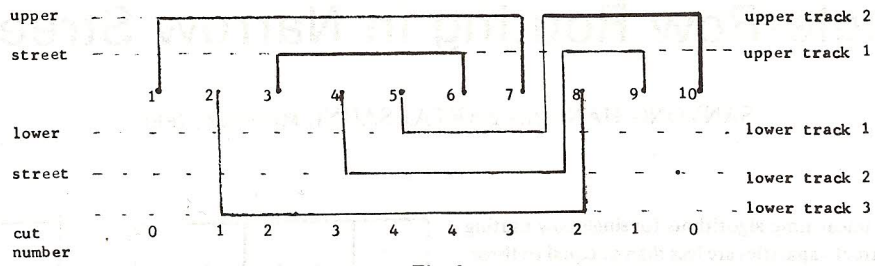


Fig. 2.

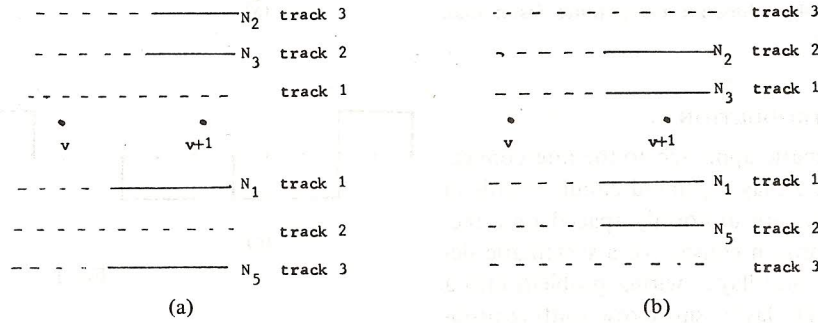


Fig. 3.

i.e., it is the *beginning* node for some net. The type *B* nodes in Fig. 2 are 1, 2, 3, 4, and 5.

(b) A *type E* node is a node at which a net *i* ends. I.e., it is a right extreme node for some net. The type *E* nodes in Fig. 2 are 6, 7, 8, 9, and 10.

(c) A *middle* or “*nonextreme*” node of a net *i* is a *type M* node. If *a* and *b* are, respectively, the *B* and *E* nodes of a net *i*, then $N_i - \{a, b\}$ is the set of middle or type *M* nodes for this net.

The *cut number* of a node is the number of nets covering that node, i.e., the number of nets between whose extreme nodes the node in question lies. The net (if any) with touch point *i* is not included in this count. Cut numbers for the nodes of Fig. 2 are given in this figure.

A *k-zone* [9] is a segment of the node axis beginning at a *B* type node with cut number *k* and ending at the next *E* type node with this cut number. Thus all cut numbers within a *k-zone* are greater than or equal to *k*. The example of Fig. 2 has the following zones:

- 0-zone: nodes 1 through 10
- 1-zone: nodes 2 through 9
- 2-zone: nodes 3 through 8
- 3-zone: nodes 4 through 7
- 4-zone: nodes 5 through 6

In general, of course, a net list may have several *k-zones* for any given *k*.

In all our algorithms, we shall attempt to obtain the net realization by processing the nodes from left to right. As in [4], each possible ordering of nets that can be encountered by a node is maintained simply as an ordered list of net indexes. There is no explicit division between the upper and lower streets. Such an explicit division is neither desirable nor necessary. Only the relative order (top to bottom) is relevant; this same order is reflected in the realization.

An advantage of this approach is that functionally equivalent situations, such as the ones in Fig. 3(a) and (b), are not treated separately.

II. $K_u = k$ AND $K_l = 1$

As noted in Section I, the layout will be constructed by scanning the nodes from left to right. Assume that the layout has been obtained upto node *i* - 1. Let (N_1, N_2, \dots, N_b) be the permutation of the nets that arrive at node *i* from the left (Fig. 4(a)). We shall show how to obtain an arrival permutation for node *i* + 1 such that this permutation will lead to a feasible layout if one exists.

If node *i* is of type *E* or *M*, then it must be a touch point for one of the nets N_j , $1 \leq j \leq b$. If it is a touch point for the net N_p and $p < b - 1$, then the routing cannot be completed with $K_l = 1$ because at least two nets (N_{b-1} and N_b) must pass below node *i* (Fig. 4(b)). If $p \geq b - 1$ and *i* is of type *M*, then the arrival permutation for node *i* + 1 is (N_1, N_2, \dots, N_b) . If $p \geq b - 1$ and *i* is of type *E*, then the arrival permutation for node *i* + 1 is obtained from (N_1, N_2, \dots, N_b) by simply deleting N_p .

When node *i* is of type *B*, we need to be somewhat careful about the construction of the arrival permutation for node *i* + 1. It is easy to see that there are at most two possibilities for this permutation (Fig. 5).

Let N_x be the net that begins at *i*. If $b > k$, then no feasible layout is possible. So, assume that $b \leq k$. When $b = 0$, (N_x) is the only permutation possible for *i* + 1. When $b > 0$, we have two possibilities: $(N_1, N_2, \dots, N_{b-1}, N_x, N_b)$ and $(N_1, N_2, \dots, N_{b-1}, N_b, N_x)$. We need to determine which of these two will lead to a layout.

Let v_b and v_x , respectively, be the end points of nets N_b and N_x . Let $v = \min\{v_b, v_x\}$. We observe that if any of the nets $\{N_1, N_2, \dots, N_{b-1}\}$ have touch points before *v*, then the layout cannot be completed and this will be detected as soon as

