COMPACT CHANNEL-ROUTING OF MULTITERMINAL NETS

M. Sarrafzadeh and F. P. Preparata*

Abstract

In this paper we describe a novel technique for solving the channel routing problem of multiterminal nets. The layout is produced column-by-column in a left-to-right scan; the number t of used tracks satisfies the bound $\delta \leq t \leq \delta+\alpha$ $(0 \leq \alpha \leq \delta-1)$, where δ is the density of the problem. The technique behaves equivalently to known optimal methods for two-terminal net problems. For a channel routing problem with C column and n nets, the algorithms run in time O(Clogn) and produce layouts that are provably wireable in three layers.

Keywords : layout techniques, channel routing problem, knock-knee layout mode, multiterminal nets, two-terminal nets, multilayer wiring.

This work was supported in part by Semiconductor Research Cooperation under Contract 83-01-035 and by the National Science Foundation under Grant MCS-81-05552.

* Coordinated Science Laboratory and Department of Electrical Engineering, University of Illinois, Urbana, IL 61801.

1. Introduction

A general two-shore channel routing problem (GCRP) consists of two parallel rows of points, called terminals, and a set of nets, each of which specifies a subset of terminals to be (electrically) connected by means of wires. The goal is to route the wires in such a way that the channel width is as small as possible. As is customary, we view a channel of width t as being on a unit grid with grid points (x,y), where both x and y are integers, with $0 \leq y \leq t + 1$ and arbitrary x. The horizontal lines are called tracks and the vertical lines columns. A vertex (x,y) of this grid at either y = 0 or y = t + 1 is a terminal; in particular, $(s_j,0)$ is a lower (or entry) terminal and $(t_i,t+1)$ is an upper (or exit) terminal. A wire is a subgraph of this grid whose edges are segments connecting adjacent vertices in the grid. A multiterminal net N is an ordered pair of (not simultaneously empty) integer sequences $((s_1,\ldots,s_k),(t_1,\ldots,t_h))$; thus, N contains lower terminals $s_1,\ldots,s_k$ and upper terminals $t_1,\ldots,t_h$. A solution to a GCRP must have, for each net, a graph on the grid that contains a path between any two terminals of that net. Note also that no two nets may share the same terminal.

We shall adopt the layout mode known as "knock-knee" [RBM, L, BB], where no two wires share an edge of the grid, but two wires may cross at a vertex or may both bend at that vertex (see Figure 1). In this mode, two distinct nets can share only a finite number of points, thereby reducing crosstalk between nets.

In the channel there is a fixed number (two or more) of conducting layers, each of which is a graph isomorphic to the channel grid. These layers are (ordered and) placed one on top of another, and contacts between two distinct layers (vias) can be made only at grid points. If two layers are

Figure 1. Illustration of the basic construct in the knock-knee mode.

connected at a grid point, no layer inbetween can be used at that grid point.

We shall use the terms "layout" and "wiring" with the following distinct technical connotations (as in [PL]).

Definition 1. A wire layout (or simply layout) for a given GCRP is a subgraph of the layout grid, each of whose connected components corresponds to a distinct net of the GCRP, in the knock-knee mode.

Notice that we can, without loss of generality, restrict ourselves to connected subgraphs which are trees, which we shall call wire-trees. (Each non-tree graph can be replaced by one of its tree subgraphs on the same set of terminals.)

Definition 2. Given a wire layout consisting of wire-trees $w_1,\ldots w_n$, a wiring is a mapping of each edge of wire-tree w_i (for $i = 1,2,\ldots,n$) to a conducting layer with vias established at layer changes.

An optimal layout of a given GCRP is a layout that uses the least possible number d of tracks. A simple-minded (and optimistic) lower bound to d can be readily established as follows.

Consider a GCRP $\eta = \{N_1,\ldots,N_n\}$, where $N_i = ((s_1^i,\ldots,s_{k_i}^i), (t_1^i,\ldots,t_{h_i}^i))$, and let $\ell_i = \min(s_1^i,t_1^i)$ and $r_i = \max(s_{k_i}^i,t_{h_i}^i)$. The interval $[\ell_i,r_i]$ represents an obvious lower bound to the horizontal track demand raised by N_i, since a terminal in column ℓ_i must be connected to a terminal in column r_i. In other words N_i is replaced by a fictitious two-terminal net N_i^* (whose two terminals may belong to the same track). We now consider the channel routing problem $\eta^* = \{N_1^*, \ldots, N_n^*\}$, and use standard methods to obtain its density δ (i.e., the maximum number of two-terminal nets which must cross any vertical section of the channel). It is clear that δ is a lower bound for the minimum number of horizontal tracks, and we call δ the <u>essential</u> <u>density</u> of the GCRP.

Two methods [BP][B] have been recently proposed for the GCRP, which use at most 2δ tracks. The methods are inherently different, but both produce the layout track-by-track; the method in [B] used exactly 2δ tracks, while the method in [BP] frequently results in more economical realizations.

In this paper we shall illustrate a method that produces the layout column-by-column (analogously to the "greedy router" of [RF]), and uses $\delta + \alpha$ tracks, where $0 \leq \alpha \leq \delta-1$.

The paper is organized as follows. In Section 2 we describe and prove the correctness of the wire layout algorithm, beginning with a systematic version that uses $2\delta-1$ tracks, and then introducing natural modifications leading to potentially simpler layouts. We also show that the method, when applied to a collection of two-terminal nets, produces a result equivalent to the one of the optimal method of Preparata-Lipski [PL]. Finally, in Section 3 we show that the obtained layouts are wireable in three layers.

2. <u>Wire layout algorithm</u>

Before describing our proposed wire layout algorithm, we examine a simpler version thereof, which uses exactly $2\delta-1$ tracks. This simpler technique will provide the intuitive background for the method; the final algorithm will be a refinement of this version.

2.1 A t=2δ−1 algorithm.

At any abscissa x we say that a net $N_m = ((s_1,\ldots,s_k),(t_1,\ldots,t_h))$ is <u>upper-active at c</u> if $t_1 \leq c \leq t_h$, and <u>lower-active at c</u> if $s_1 \leq c \leq s_k$; N_m is <u>active at c</u> if it is both upper-active and lower-active at c. For ane x_0 in the interval (c,c+1], $\min(s_1,t_1) \leq c < \max(s_k,t_h)$, a vertical line $x=x_0$ cuts N_m in at least one point. Each intersection of N_m with $x=x_0$ identifies a <u>strand</u> of N_m at x_0.

The invariant (and the central feature) maintained by the algorithm is the following :

<u>Property 1</u>. Each net upper-active at c has a strand lying <u>above</u> a strand of any other lower-active net; each net lower-active at c has a strand lying <u>below</u> a strand of any other upper-active net.

If this property holds, then the layout of the column is straightforward: Indeed, if $c = s_p^j = t_q^i$ (i.e., column c has an entry terminal of net N_j and an exit terminal of net N_i), by Property 1 there is a strand σ_i of N_i lying above a strand σ_j of N_j; thus we connect s_p^j to σ_j and t_q^i to N_i by means of nonoverlapping vertical wires (see Figure 2).

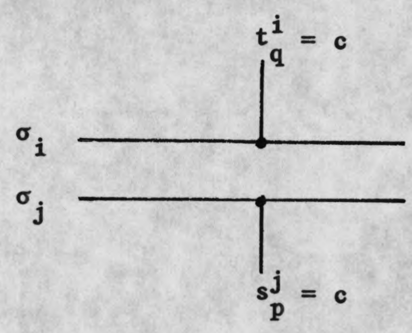


Figure 2. Column layout

Property 1 will be readily established if we algorithmically guarantee the following (specialized) invariant:

<u>Property 2</u>. Let J be the set of indices of the nets of η, and let σ: J -> $\mathbf{Z}^+$ be a function from the set J to the positive integers. At x = c, each net N_m upper-active at c has a strand at y = $\sigma(m)$; each net N_m lower-active at c has a strand at y = $-\sigma(m)$. This means that if N_m is active at c, it has a symmetric pair of strands at y = $\sigma(m)$ and y = $-\sigma(m)$[(1)] (Figure 3).

For convenience, in the GCRP statement, a net will be represented by a tree as in Figure 4. Suppose we display all the members of $\eta = \{N_1,\ldots,N_n\}$ each as in Figure 4, in the correct vertical alignment. For a given column c, we cut a vertical slice $[c-\varepsilon, c+\varepsilon]$,$0<\varepsilon<1$, and retain only the net fragments

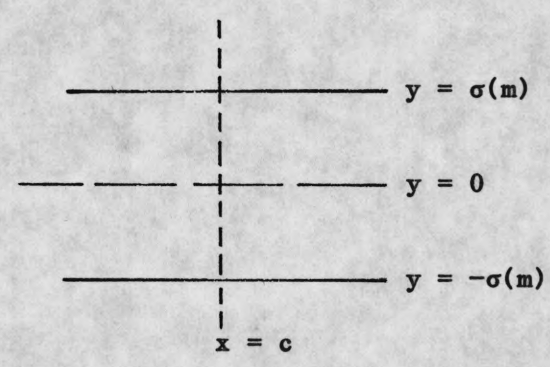


Figure 3. The two symmetric strands of N_m active at c.

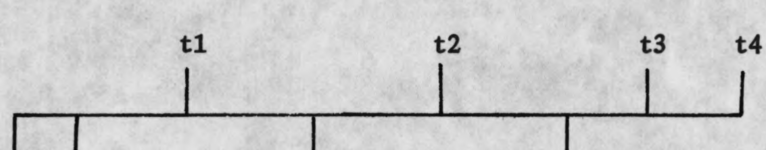


Figure 4. Representation of a multiterminal net in GCRP statement.

---

(1) Thus, by convention, track y=0 is not used. Later, we shall see that the strands (of the same net) at y=+1 and y=−1 can be made to coincide.

containing a terminal (at most two): this yields the <u>column state</u> (state(c)), i.e., the layout requirement of column c. The 20 different possible states are shown in Figure 5 (where | denotes "empty", and "↑" denotes "trivial", i.e., a two-terminal net with $s_1 = t_1 = c$). For a column c, let $d_L(c)$ and $d_R(c)$ be the local densities of the problem in the open intervals (c-1,c) and (c,c+1), respectively. With reference to a left-to-right scan, we say that c is a density increasing column (d.i.c.) if $d_L(c) < d_R(c)$, and is a density decreasing column (d.d.c.) if $d_L(c) > d_R(c)$. With this definition, the column states are readily classified as in Figure 5 as d.i. (density increasing), d.d. (density decreasing) and d.p. (density preserving). Hereafter, <u>a terminal will be labeled with the index of the net to which it belongs</u>.

We begin by considering the d.p. columns. States ┬, ┴, ┼ and ╪ are readily handled, as shown in Figure 2 (or in a trivial variant thereof). So, we must consider states ┘┌ and ┐└. Here N_i terminates and N_j begins. The two nets N_i and N_j can be concatenated to form a <u>run</u> of nets. The transition between two nets of the same run can be handled very simply. In either of the cases illustrated in Figure 6, we assign to N_j the same track(s) assigned to N_i to the left of c, and Property 2 is maintained.

We now give a less informal description of the handling of a d.p. column. Specifically, here and hereafter, a <u>right bend</u> is a layout construct

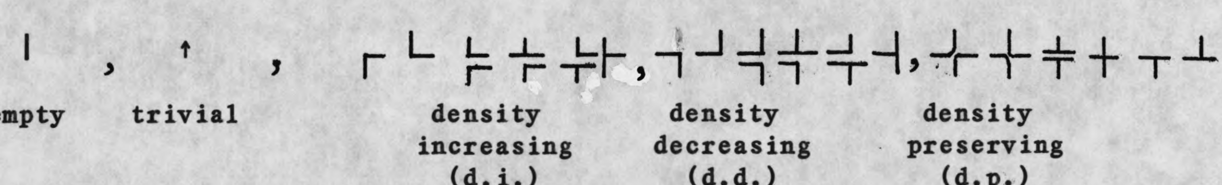


Figure 5. Possible column states.

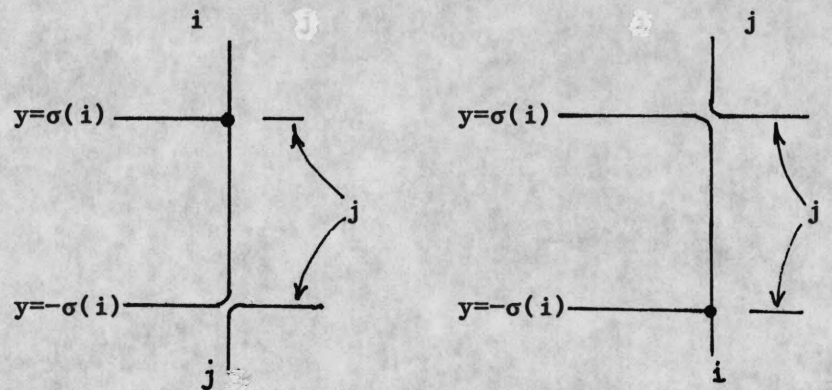


Figure 6. Handling of a density-preserving column

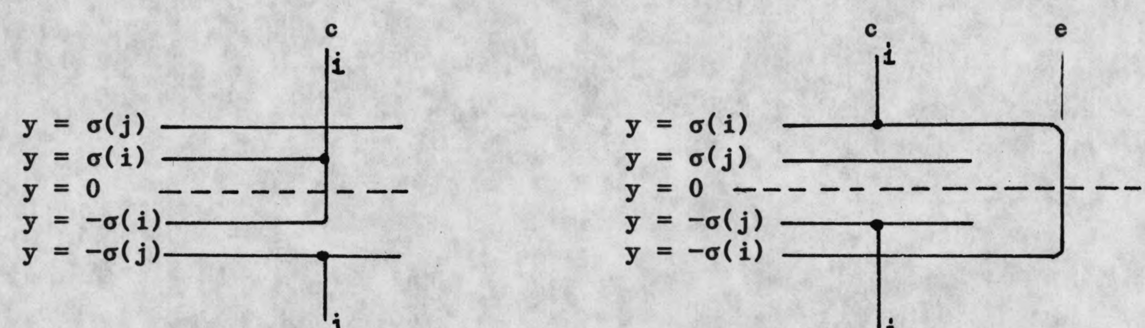
of the types "┐" or "┌", whereas a <u>left</u> <u>bend</u> is one of the constructs "┘" or "└". Available is an array describing the function σ, and a priority queue Q of the available tracks in the set $\{y = i \mid i=1,...,\delta\}$. In the following subroutine ,LAYOUT D.P.C.(u,ℓ,c), u and ℓ are respectively the names of the nets having an upper (exit) or lower (entry) terminal in column c. Of course, either u or ℓ (or both) may be equal to Λ, (the index of the empty net); in which case, any "connect" operation involving the empty net is void.

```
     PROCEDURE LAYOUT-D.P.C (u,ℓ;c)
BEGIN IF state(c) ∈ {┬, ┴, ┼, ╪} THEN connect upper terminal to
        y = σ(u) and lower terminal to y = −σ(ℓ)
     ELSE BEGIN
            r := net starting at c;
            IF state(c) = ┤ THEN
               BEGIN connect lower terminal to y = σ(ℓ) and y = −σ(ℓ);
                     σ(r) := σ(ℓ);
                     connect upper terminal to σ(r) with left bend
               END
            ELSE
               BEGIN connect upper terminal to y = σ(u) and y = −σ(u);
                       σ(r) := σ(u);
                     connect lower terminal to −σ(r) with right bend
               END
          END
END
```

By forming runs of nets and handling them as shown in Figure 6, we are effectively partitioning the channel into <u>blocks</u> of contiguous columns, so that changes of density occur only at columns separating adjacent blocks. The preceding discussion gives us the following result.

<u>Lemma 1</u>: Inside each block the set of tracks used remains fixed.

We now consider the handling of d.d. columns, with the assumption (to be substantiated later) that a terminating net has at most two disconnected strands in tracks above and below y = 0. Recall that at a d.d. column one or two nets terminate. Suppose, at first, that just one net terminates (states ┐, ┘, ┤, ┼, ┼). States ┐, ┘, and ┤ are handled in a straightforward manner, by connecting both strands of the net to the appropriate terminal. There remain states ┼ and ┼, of which we just need to consider ┼ (the other case being handled symmetrically). Referring to Figure 7, let N_i be the terminating net and N_j be the continuing net. If $\sigma(j) > \sigma(i)$, then the termination of N_i is straightforward (Figure 7a). If $\sigma(j) < \sigma(i)$, then N_i cannot be terminated at column c. Thus we extend both strands of N_i to the closest d.i.



(a) (b)

Figure 7. Handling of a density-decreasing column c; N_i is the terminating net.

or empty column e to the right of c. If e is an empty column, then the two strands of N_i are connected at e in a straightforward manner (Figure 7b). When e is a d.i. column, the operation of splicing of the two strands of N_i at e will be considered later in connection with the layout of a d.i. column. When two nets terminate at c (state(c) = ╡), then one net is spliced at c (as N_i in Figure 7a) and the other is extended (as N_j in Figure 7b). We shall make use of an integer parameter EX, which denotes the number of nets being extended to the right of the current column c.

The preceding discussion is formalized in the subroutine LAYOUT - D.D.C(u,ℓ; c,EX); this subroutine makes use of the priority queue Q of the available tracks (only positive ordinates), with the usual notations "Q ⇐ " (add to Q) and " ⇐ Q" (extract from Q), and, of a priority queue P of the extended nets.

```
        PROCEDURE LAYOUT - D.D.C.(u,ℓ; c; EX);
  BEGIN Connect upper terminal to y = σ(u) and lower terminal to y = -σ(ℓ);
        r := net continuing in column c;
        IF state(c) ∈ {┐, ┘, ┤} THEN
             BEGIN Q ⇐ σ(r);
                   connect y = σ(r) and y = -σ(r);
             END
        IF state(c) ∈ {╪, ╤} THEN
             BEGIN e :=  net continuing at column c;
                  IF σ(r) < σ(e) THEN
                      BEGIN Q ⇐ σ(r);
                            connect y = σ(r) and y = -σ(r);
                      END
                  ELSE BEGIN EX = EX + 1;
                            P ⇐ r;
                      END
             END
        IF state(c) = ╡ THEN
             BEGIN EX = EX + 1;
                  IF σ(u) > σ(ℓ) THEN
                      BEGIN connect y = σ(ℓ) and y = -σ(ℓ);
                            P ⇐ u;          (*extended net *)
                            Q ⇐ σ(ℓ);   (*available tracks *)
                      END
```

```
ELSE BEGIN connect y = σ(u) and y = −σ(u);
           P ⇐ ℓ ;
           Q ⇐ σ(u);
     END
  END
END
```

<u>Lemma 2</u>: Any d.d. column can be processed without increasing the number of tracks.

<u>Proof</u> : When any of column states ┐, ┘ ,┤ is processed, two tracks become free. For column state = , two tracks become free and one net is extended, and finally, for column states ┴┐ and ┘┬ either two tracks are freed or one net is extended, depending on the relative positions of the tracks carrying the terminating and the continuing nets. So, no tracks are added while a d.d. column is being processed.

We now consider the handling of d.i. columns. Recall that at a d.i. column one or two nets begin. First suppose that just one net N_i begins (states └ ,┌ ,├ ,└┬,┴┌). States └ , ┌ , ├ are handled in a straightforward manner: if there is no extended net, by assigning two unused tracks to N_i; otherwise by connecting strands of an extended net together and running N_i on the tracks previously used by the extended net. There remain states └┬ and ┴┌, of which we just consider └┬. Referring to Figure 8, let N_i be the new net, N_j be the continuing net and N_h be an extended net (if any). If $\sigma(j) > \sigma(i)$, then all the connections are straightforward and N_i is assigned the tracks previously used by N_h (Figure 8a). If $\sigma(j) < \sigma(i)$, then N_j cannot be connected to $y = -\sigma(j)$, so we shall run N_j on $y = -\sigma(i)$ temporarily (in the channel columns where N_i is not lower-active). The two tracks $y = -\sigma(i)$ and $y = -\sigma(j)$ are connected at the closest column e to the right of c with the following property: the lower terminal is labeled either i, or j, or Λ (Figure

9). (Note that between columns c and e net N_j has <u>three</u> strands.) Finally, when two nets start at c (state(c) = ⊨), if there is just one or no extended net, then the connections to upper (u) and lower (ℓ) terminal are straightforward as in Figures 8a and 8b. If instead there are two or more extended nets, then only one will be connected at column c and the others will be further extended to the right. The integer parameter EX is updated (decreased) as the extended nets are connected. The preceding discussion is formalized in the subroutine LAYOUT - D.I.C. (u, ℓ ; c; EX). This subroutine makes use of addi-

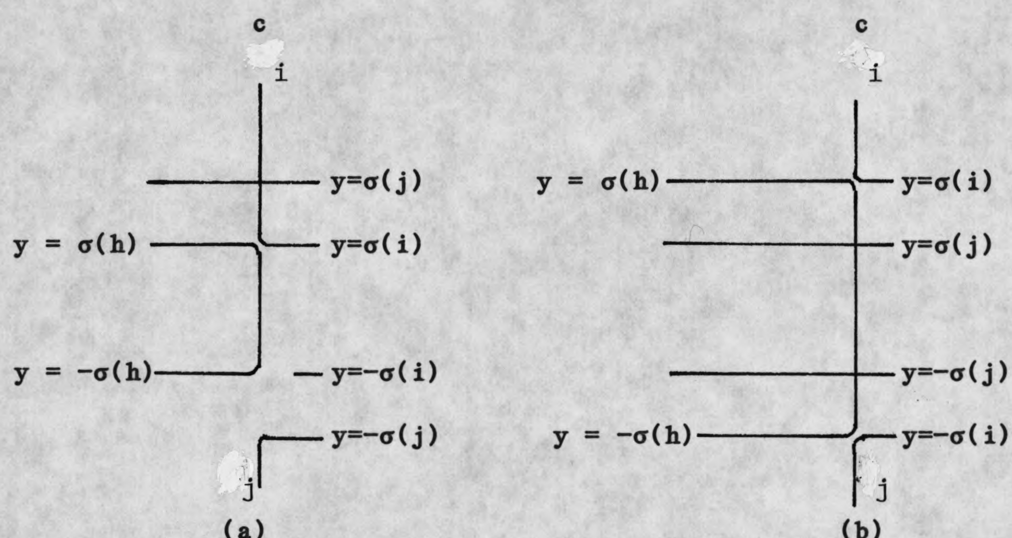


Figure 8. Handling of a density-increasing column; N_i is the beginning net, and N_h is an extended net.

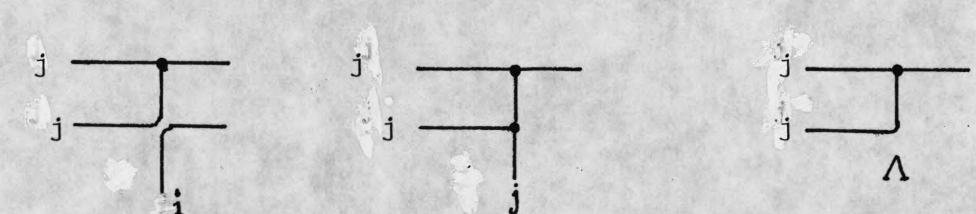


Figure 9. Temporary extension of a net on the inactive track of a beginning net.

tional data structure to handle nets temporarily assigned two tracks in either (lower or upper) half-channel. Specifically, for the upper portion of the channel, we shall use two binary search trees AU and AU^{-1} (and analogously, AL and AL^{-1} for the lower portion of the channel). Both AU and AU^{-1} are organized on the basis of the index of a net, and support operations of MIN, MEMBER, INSERT, and DELETE in time logarithmic in their size. Their function is the following: AU(t) = s (i.e., search of AU for member t) means that the third track used by t is σ(s); conversely, $AU^{-1}(t) = s$ means that track σ(t) is used, as a third track, by s.

```
PROCEDURE LAYOUT - D.I.C. (u, ℓ , c; EX)
f := 0;
BEGIN IF EX ≠ 0 THEN
      BEGIN r ⇐ P;                (* P is the queue of the extended nets *)
            EX := EX - 1;
            Connect σ(r) and -σ(r);
            t1 := σ(r);                  (* t1 is a track made available *)
            f := 1;    (* f=1 means that the column is occupied by a wire
                          between y=σ(r) and y=-σ(r) *)
      END
      ELSE t1 ⇐ Q;          (* Q is the queue of available tracks *)
      IF state(c) ∈ {┌ , └ , ├ , ┼ , ╪ } THEN
         BEGIN IF u ≠ Λ AND state(c) ≠ ┼ THEN σ(u) := t1;
               IF ℓ ≠ Λ AND state(c) ≠ ╪ THEN σ(ℓ) := t1;
               IF state(c) = ┼ THEN
                  BEGIN IF f = 0 OR (f = 1 AND σ(u) > t1) THEN t1 := σ(u);
                        IF f = 1 amd σ(u) < t1 THEN
                           BEGIN AU(u) := ℓ;
                                 AU^-1(ℓ) := u;
                           END
                  END
               IF state(c) = ╪ THEN
                  BEGIN IF f = 0 OR (f=1 AND -σ(ℓ)<-t1) THEN -t1 := -σ(ℓ);
                        IF f = 1 AND -σ(ℓ) > -t1 THEN
                           BEGIN AL(ℓ) := u;
                                 AL^-1(u) := ℓ;
                           END
                  END
               Connect upper terminal to t1 with a left bend;
               Connect lower terminal to -t1 with a right bend;
         END
      ELSE BEGIN σ(u) := t1;                    (*state(c) = ╞ *)
               connect upper terminal to t1 with a left bend;
               t2 ⇐ Q;
```

```
                σ(ℓ) := t2;
                IF f = 1 AND t2 < t1 THEN
                   BEGIN AL(ℓ) := u;
                         AL^-1(u) := ℓ;
                         t2 := t1;
                   END
                ELSE  Connect lower terminal to -t2 with a right bend;
          END
END
```

<u>Lemma 3</u>: Any d.i. column c can be processed using no more than 2δ tracks.

<u>Proof</u>: It is sufficient to show that at a d.i. column c: (1) no tracks are added if $d_R(c) \leq d_m(c)$ (recall that $d_R(c)$ denotes the local density in (c, c+1)), and $d_m(c) = \max_{i<c} d_R(i)$; (2) Two (symmetric) tracks are added if $d_R(c) = d_m(c) + 1$ (this also implies that $d_R(c-1) = d_m(c)$); (3) four tracks are added if $d_R(c) = d_m(c) + 2$ (this also implies that $d_R(c-1) = d_m(c)$).

Indeed, if $d_R(c) = d_m(c) + 1$ or $d_R(c) = d_m(c) + 2$, new (symmetric) tracks are added and the column is laid out in a straightforward manner (as had been described in the handling of d.i. columns). Instead, if $d_R(c) \leq d_m(c)$ then as described earlier, states ┌, └, and ├ are trivially handled by connecting the new net to a free track (or a track occupied to the left of c by an extended net). States ┴┌, └┬ and └┌ are also handled trivially by running the new net(s) on a free track (or tracks occupied to the left of c by an extended net).

The preceding discussion reveals that at a column c with $d_R(c) = \delta$ ($\delta = \max_i d_R(i)$) the number of occupied tracks is 2δ and the number of tracks never exceeds 2δ in any column i for all i > c.

The layout procedure scans the channel column by column from left to right. It calls the appropriate subroutines according to the state of the current column. We shall formalize this in the subroutine LAYOUT1.

```
        PROCEDURE LAYOUT1;
    BEGIN c := 1;                                        (* first column *)
        EX := 0;                                         (* no extended nets *)
        WHILE EX ≠ 0 or there is any d.i.c. left DO
            u := upper terminal in column c;
            ℓ := lower terminal in column c;
            IF AU ≠ ∅ THEN
               BEGIN IF AU^-1(u) ≠  Λ THEN
                  BEGIN s := AU^-1(u);    (* third strand of s on σ(u) *)
                        connect y = σ(s) and y = σ(u);
                        connect u to y=σ(u) with left bend;
                        delete AU^-1(u) and AU(s);
                  END
                  IF AU(u) ≠ Λ THEN
                     BEGIN s := AU(u);        (* third strand of u on σ(s) *)
                           connect u to y = σ(s) and y = σ(u);
                           delete AU(u) and AU^-1(s);
                     END
                  IF u = Λ THEN
                     BEGIN t := min AU;
                           s := AU(t);
                           connect y = σ(s) to y = σ(t);
                           delete AU(t) and AU^-1(s);
                     END
               END
            IF AL ≠ ∅ THEN   (* analogous to the above when AU ≠ ∅ *)
            IF state(c) = ↑ THEN connect upper and lower terminals;
            ELSE IF c = d.p.c. THEN call LAYOUT - D.P.C. (u,ℓ; c);
            ELSE IF c = d.d.c. THEN call LAYOUT - D.D.C. (u,ℓ; c; EX);
            ELSE IF c = d.i.c. OR empty THEN call LAYOUT-D.I.C.(u,ℓ; c; EX);
            c := c + 1;
        END
    END.
```

<u>Theorem 1</u>. Any GCRP with essential density δ, can be laid out in $2\delta - 1$ tracks using the LAYOUT1 algorithm.

<u>Proof</u>: Referring to Lemmas 1, 2 and 3, it can be noted that only at d.i. columns we add tracks. We have also shown in Lemma 3 that the number of occupied tracks is twice the essential density, namely 2δ. A closer look at the algorithm (LAYOUT1) reveals that tracks 1 and -1 are always occupied by the same net (property 2) so we can merge them. As a result, only $2\delta-1$ tracks are required to lay out any GCRP with essential density δ.

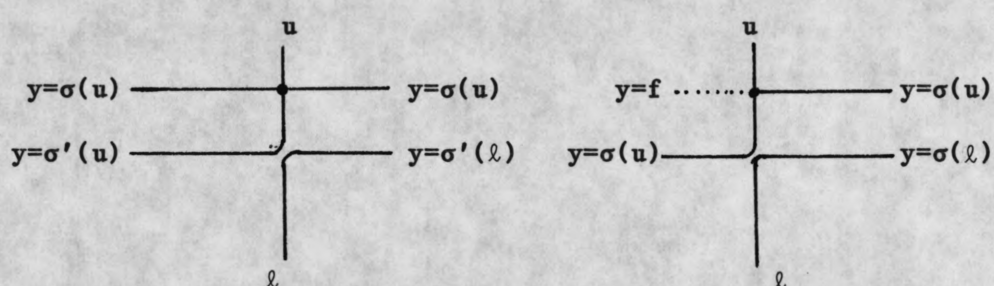
2.2 An Improved Algorithm

The algorithm we have presented in the last subsection was intended to provide intuition for a possible solution to any GCRP and to establish an upper-bound to the number of used tracks. The main feature of this algorithm is that, in conjunction with heuristics, it can result in a rather efficient and yet provably good solution to any GCRP. The first thing to note is that the second strand of a net should be added only when it is necessary: indeed, it might very well happen that for some nets we never need to add the second strand. Second, it is only natural to splice the split nets (nets having more than one strand) as soon as it is feasible in a left to right scan. We shall name the improved technique "LAYOUT2". We shall see later that in this method the two strands of a net may be no longer symmetric, so we shall use the notation $\sigma'(t)$ to denote the track used by the second strand of a net t (as opposed to $-\sigma(t)$ in "LAYOUT1"), with the convention that $\sigma'(t) < \sigma(t)$.

We now look at the handling of the column states by "LAYOUT2". The notation of inclusion between states of a column is defined in a natural way, e.g., ┐ ⊆ ┐└. A column s is said to be a T-column if ┬, ┴ ⊆ state(s) (states ┬, ┴, ┴/┌, └/┬, ┘/┬, ┴/┐, ┴/┬ are T-states). Handling of any non-T-column is straightforward, as shown in the discussion of "LAYOUT1", as in the handling of states ┬ and ┴. So, we need to consider states ┴/┌, └/┬, ┘/┬, ┴/┐ and ┴/┬.

We begin with the two d.i. T- columns (┴/┌, └/┬), of which we just need to consider ┴/┌ (the other case being handled symmetrically). If the net N_u, to be connected to the upper terminal u, has two strands, then we can connect $y = \sigma(u)$ and $y = \sigma'(u)$ and run the beginning net N_ℓ on $y = \sigma'(u)$ as shown in Figure 10a. If instead, N_u has only one strand but there is a free track at $y = f$, with $f > \sigma(u)$, then we can connect $y = \sigma(u)$ to $y = f$ and continue N_u on track f, while N_ℓ is assigned track $\sigma(u)$ (Figure 10b). Other variations are being handled trivially.

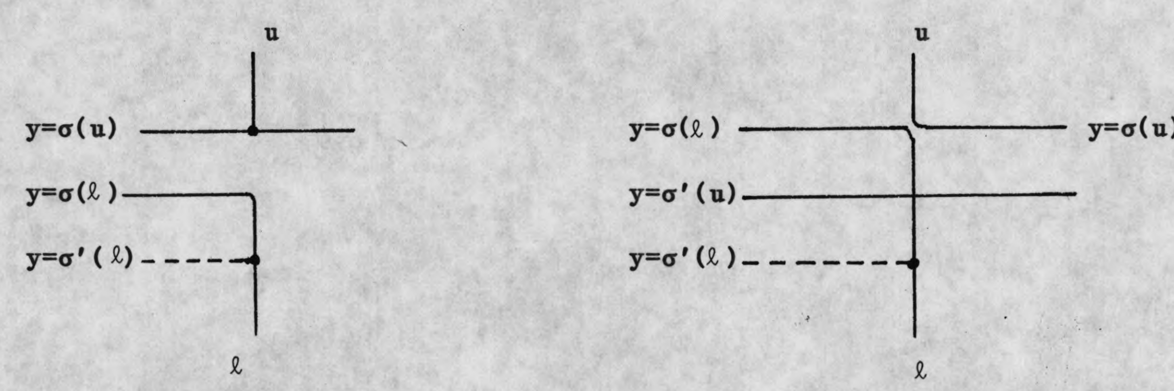
Next we shall turn our attention to the handling of the two d.d. T-columns (╤, ╤) of which we just need to consider ╤ (the other case will be handled symmetrically). The policy is to connect the lower terminal ℓ to $y = \sigma(\ell)$ (and also to $y = \sigma'(\ell)$ if it exists) and connect the upper terminal u to $y = \sigma(u)$ if $\sigma(u) > \sigma(\ell)$ (Figure 11a), or to $y = \sigma(\ell)$ with a left bend, otherwise (Figure 11b). Other variations are being handled trivially.



a. u has more than one strand

b. $y=f > y=\sigma(u)$

Figure 10. Handling of a d.i. T-column



a. $\sigma(u) > \sigma(\ell)$

b. $\sigma(u) < \sigma(\ell)$

Figure 11. Handling of a d.d. T-column

It remains to consider the handling of a d.p. T-column ($\frac{\perp}{\top}$). Denoting, as usual, by N_u and N_ℓ the two nets having the upper and lower terminals, respectively, there are eleven possible cases (ignore trivial mirror symmetries), depending upon the number of strands of N_u and N_ℓ and their relative positions. The handling of these cases is illustrated in Figure 12, and deserves no further comment.

The preceding improvements yield the following result:

<u>Theorem 2</u>: Any GCRP with essential density δ can be routed in $\delta + \alpha$ tracks for $0 \leq \alpha \leq \delta-1$ by the "LAYOUT2" algorithm.

While δ is a lower bound for the minimum number of horizontal tracks, we have shown an upper bound of $2\delta-1$ for the number of tracks required by any

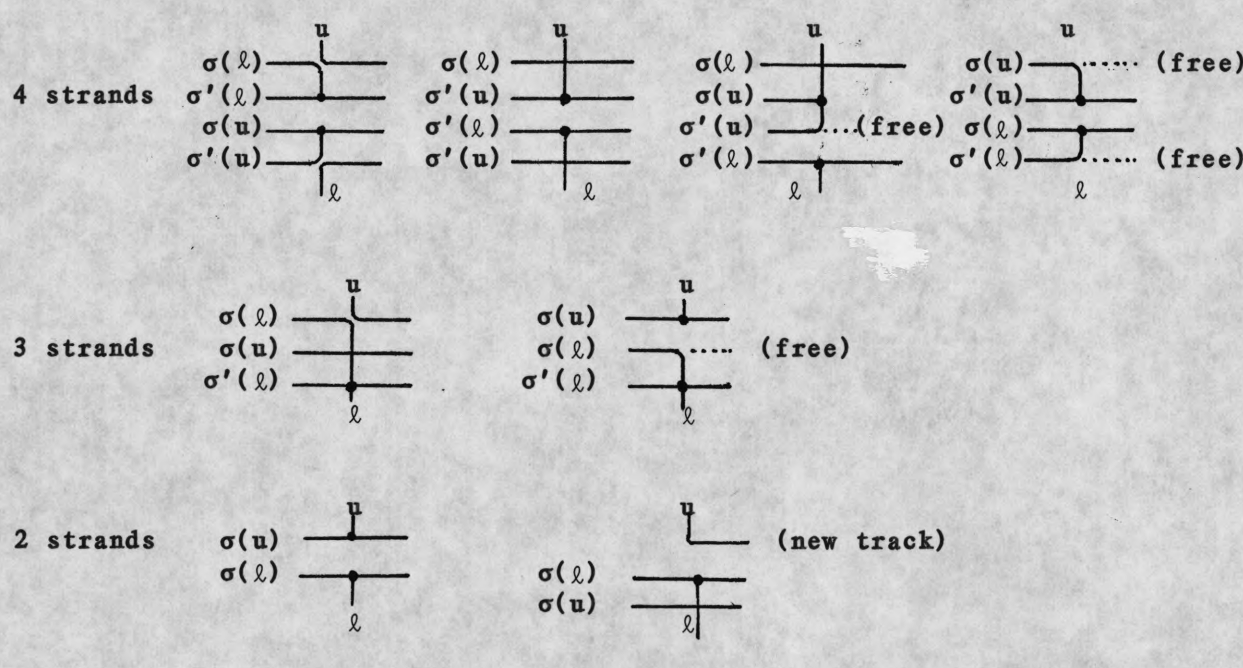


Figure 12. Handling of a d.p. T-column

GCRP using the "LAYOUT1" algorithm. By saving extra tracks (beyond δ) in the "LAYOUT2" algorithm we frequently can solve any GCRP in fewer than 2δ-1 tracks. Simulations reveal that α, the number of tracks used beyond δ, is rather small. In fact for some GCRP α is equal to zero.

Next we shall turn our attention to a special case of GCRP, namely a two-terminal net CRP. We shall route a CRP by the "LAYOUT2" algorithm. Of the twenty possible column states, only the following ten states (|, ↑, ┐, ┘, ┤, └, ┌, ├, ┐└, ┘┌) may arise. Recall that the algorithm scans the channel from left to right and processes successively all the columns. Note that Property 1 holds in all columns except when state(c) = ┤. The processing of this column is straightforward if σ(u) > σ(ℓ) (where ,as usual, u is the upper terminal and ℓ the lower terminal) as in Figure 13a. If, on the other hand, σ(u) < σ(ℓ), then we shall connect the upper terminal u to y = σ(u) and connect the lower terminal to y = σ'(ℓ) where we have assigned σ'(ℓ) ← σ(u) as in Figure 10b.

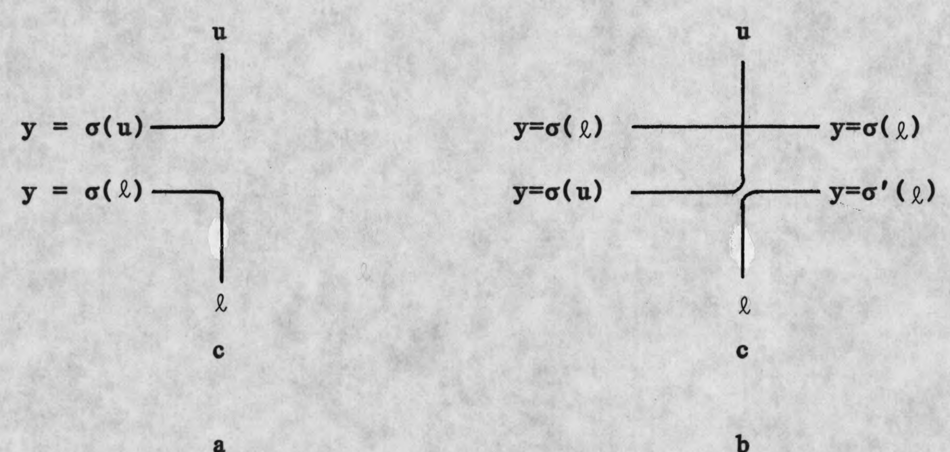


Figure 13. Layout cases for the two-terminal net CRP.

We shall make use of a priority queue P (FIFO) of the extended nets. So N_ℓ is introduced into the queue (Figure 13b); but in the interval (c,e), where e is the closest density-increasing or empty column (states | , └ , ┌ , └┌) to the right of c, the number of occupied tracks will not exceed the local density $d_L(c)-2 \leq \delta-2$, so that we can connect the two strands of the extended N_ℓ at column e (Figure 14).

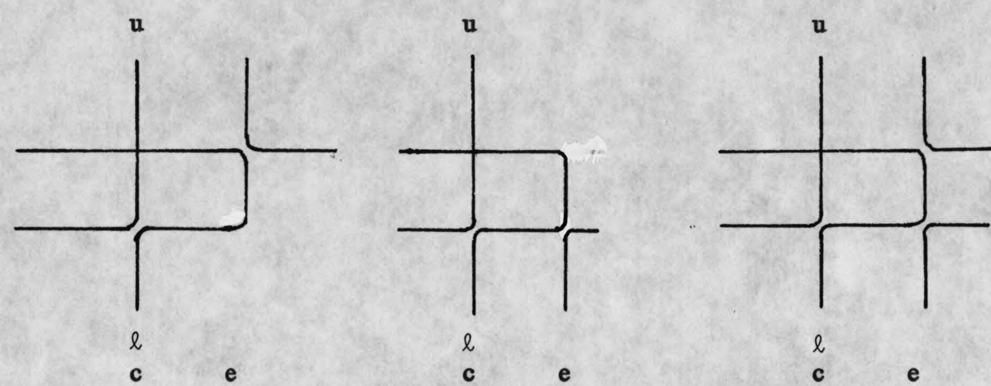


Figure 14. Layout of extended nets in the two-terminal net CRP.

The preceding discussion gives us the following result:

<u>Theorem 3</u> Any two terminal net CRP with density δ can be routed using δ tracks by the "LAYOUT2" algorithm.

3. <u>Three-layer wireability</u>

We now return to the general problem and show that any layout produced by "LAYOUT2" can be wired with three layers without increasing the number of tracks used in the layout phase.

Our arguments are heavily based on the wireability theory developed by Preparata and Lipski [PL], to which the reader is referred. We begin by observing that in the layout of multiterminal nets there is one more type of grid points, the "┬", in addition to the five standard ones encountered in the layout of two-terminal nets(the "crossing", the "knock-knee", the "bend", the "straightwire", the "empty"); these six types are illustrated in Figure 15. Following the arguments in [PL], suppose that in a given layout W we replace each non-knock-knee grid-point by a crossing: it is very simple to show that if the resulting layout W^* -- a "full" layout -- can be wired with three layers, so can the original W.

Note that in the transformation from W to W^* we have ,in essence, obtained a two-terminal net problem. We must only verify that it satisfies the three-layer-wireability sufficient condition proposed by Preparata and Lipski:

We begin by observing the following facts :

(1) at a d.p. column (states ┐└ and ┘┌) we have at most one knock-knee.

(2) at a d.d. column (states ┴┬ , ┴┐ ,and ┤┐) we have at most one knock-knee.

(3) at a d.i. column we have either one knock-knee (states ┌ , └ , ┴┌ , └┬) or two (states ├┌ and ├) if the priority queue of the extended nets is not empty.

(a) (b) (c) (d) (e) (f)

Figure 15. Types of grid-point in multiterminal net layout :
(a): T, (b): crossing, (c): knock-knee; (d): bend;
(e): straight wire; (f): empty.

We can therefore assume that every column contains one of the following: (i) a single knock-knee of the form ─┘┌─, (ii) a single knock-knee of the form ┤┌, or (iii) two knock-knees, of which the lower one is of the form ─┘┌─ and the upper one is of the form ─┐└─. As is customary, we graphically denote a knock-knee grid point by means of a $\sqrt{2}$ -length diagonal, centered on the grid point and crossing both wires as shown in Figure 16.

Figure 16. Graphical equivalents (diagonals) of knock-knees.

With this convention we can convert any layout to a "diagonal diagram". We augment the diagram by adding dummy tracks 0 and $\delta + \alpha + 1$, placing ⧄ (knock-knee of the form ─┐└─) on track $\delta + \alpha + 1$ for every column of type (i), and placing ⧅ (knock-knee of the form ─┘┌─) on track 0 for every column of type (ii). We shall refer to these additional diagonals as <u>dummy</u> diagonals. In this way we can restrict ourselves to the case where all nonempty columns are of type (iii).

Preparata and Lipski have shown that any diagonal diagram of this type corresponds to a full layout that is wireable with three layers, provided that we allow for layout modifications in correspondence with special pattern of diagonals. We now show that, in the layouts produced by our algorithms, such patterns never arise.

A representative of the special diagonal pattern is shown in Figure 17a, together with the corresponding fragment of layout. This layout shows that two disconnected strands of a net have been spliced in column c+1 and state(c) is either ─┐└─ (Figure 17a) or └┌ (Figure 17c). In either case, our policy to splice a two-strand net as soon as possible, will generate the layouts shown in Figures 17b,d (i.e., one extended net is spliced at column c rather than c+1), so that the cases shown in Figures 17a,c never arise.

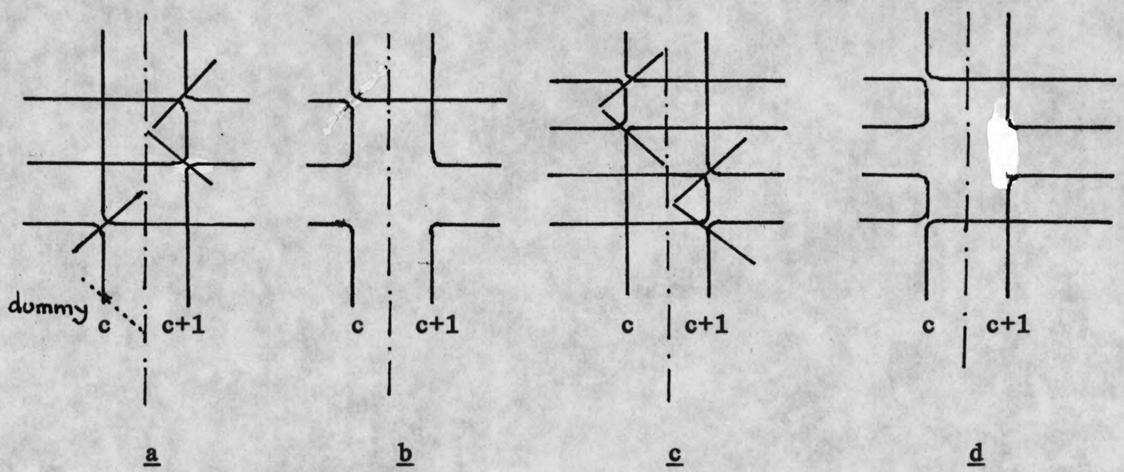


Figure 17.

Thus, as an immediate consequence of the above obsevations and of the results established in [PL], we conclude :

Theorem 4 Any layout produced by the "LAYOUT2" algorithm that uses $\delta + \alpha$ tracks (for $0 \leq \alpha \leq \delta-1$) can be wired in three layers using $\delta + \alpha$ tracks.

Since both the layout and the wiring algorithms scan through the channel column-by-column, they could run simultaneously.

To illustrate the method, we now give two examples : a GCRP (Figure 18), where $\delta = 5$ and $\alpha = 1$, and a two-terminal net CRP (Figure 19), with $\delta = 8$ and $\alpha = 0$.

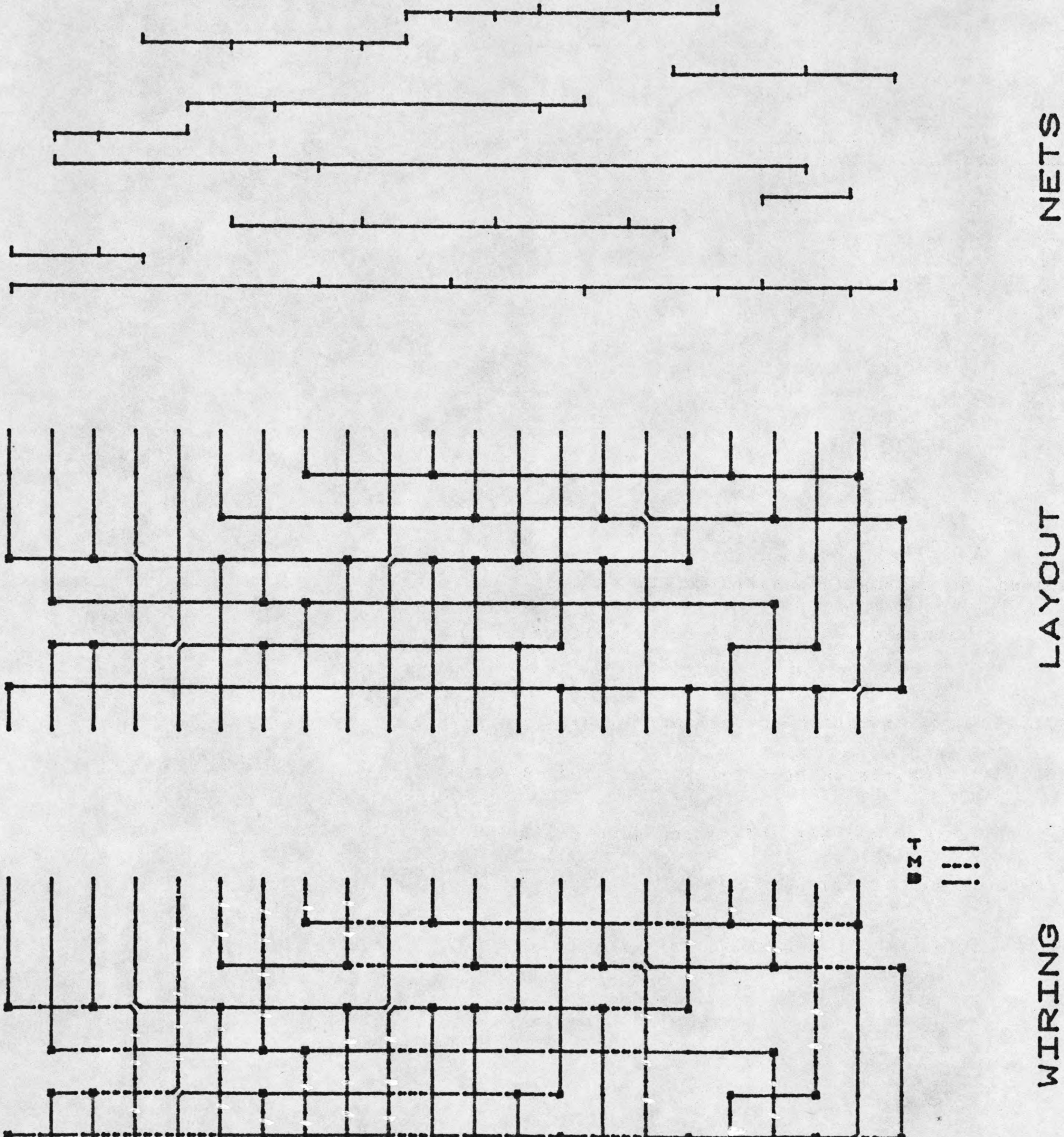


Figure 18

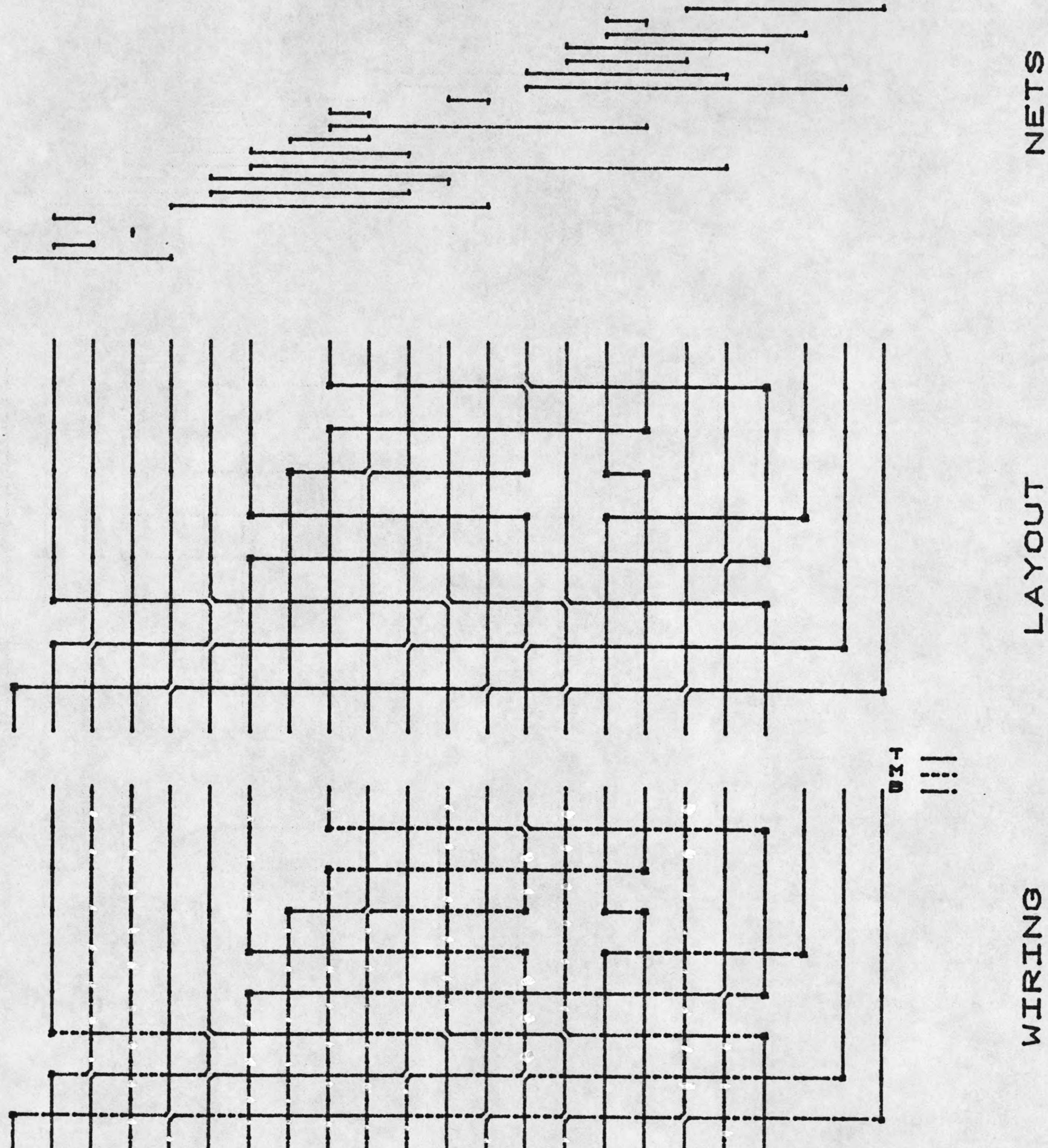


Figure 19

Before closing the section, we briefly analyze the time performance of the proposed algorithm LAYOUT 2. Denoting by C the number of columns, the input is assumed to be in the form of a sequence of pairs $((u_i, \ell_i) \mid i=1,\ldots,C)$ where $u_i, \ell_i \in \{1,\ldots,n\}$ specify the upper and lower terminals of column i; these pairs are stored in an array. Processing of each column i involves some actions on the data structure required by the procedures described earlier (priority queues Q and P, search trees AU, AU^{-1}, AL, AL^{-1}, and the array σ). Inspection of the algorithms reveals that the number of operations performed at each column is bounded by a constant; in addition , each of these operations takes time at most logarithmic in the size of the data struture (which is O(n)). Thus we conclude :

Theorem 5 "LAYOUT2" run in time proportional to Clog n, where C is the number of columns and n is the number of nets.

It is relatively straightforward to show that also wiring can be accomplished within the same time bound.

<u>References</u>

[B] Baker, B. S., private communication

[BB] Bolognesi, T. and D. J. Brown, "A channel routing algorithm with bounded wire length," draft.

[BP] Brown, D. J. and F. P. Preparata, "Three-layer channel routing of multiterminal nets," draft

[L] Leighton, F. T., "New lower bounds for channel routing," draft, 1981.

[PL] Preparata, F. P. and W. Lipski, Jr., "Optimal three-layer channel routing," <u>Proc. 23rd IEEE Symposium on Foundations of Computer Science</u>, Chicago, IL, Nov. 1982, pp. 350-357.

[RBM] Rivest, R. L., A. Baratz and G. Miller, "Provably good channel routing algorithms," <u>Proc. 1981 CMU Conference on VLSI Systems and Computations</u>, Oct. 1981, pp. 153-159.

[RF] Rivest, R. L. and M. Fiduccia, "A greedy channel router." <u>Proceedings 19th IEEE Design Automation Conference</u>, 1982, pp. 418-424.