Algorithmic Aspects of VLSI Layout
Eds. M. Sarrafzadeh and D. T. Lee

SWITCH-BOX ROUTING UNDER THE TWO-OVERLAP WIRING MODEL

TEOFILO F. GONZALEZ, SHASHISHEKHAR KURKI-GOWDARA
Department of Computer Science, University of California, Santa Barbara, CA 93106, USA

and

SI-QING ZHENG
Department of Computer Science, Louisiana State University Baton Rouge, LA 70803, USA

ABSTRACT

We present a new algorithm for the switch-box routing problem under the two-overlap wiring model. An instance of this problem is given by a rectangle R, and a set N of nets each formed by two terminal points located on the boundary of R. The objective is to interconnect the points in each net by a set of wires inside R on k layers that satisfy the constraints in the two-overlap wiring model. In this wiring model at most two wires may be assigned to the same track or column unit segment in all the layers. Given any two-overlap wirable instance, our algorithm adds at most two tracks and two columns and wires it in seven layers. The time complexity for our algorithm is $O(n)$ when the set of n terminal points is initially ordered.

Keywords: Switch-box routing, two-overlap, efficient algorithms, layer assignment.

1. Introduction

Channel routing (*CR*) and *switch-box routing* (*SBR*) are fundamental detailed routing problems in computer-aided design of VLSI layout systems. An *SBR* problem instance consists of a rectangle R and a set of *signal nets*, $N = \{n_1, n_2, \ldots, n_m\}$, where the n_i's are mutually disjoint sets of *terminal points* located on the boundary of rectangle R. The objective is to construct a wiring in R, whenever one exists. In the *CR* problem the terminal points are located on two opposite sides of R, and the objective is to construct a minimum *channel width* wiring.

The *CR* and *SBR* problems under the non-overlap wiring models such as the *Manhattan model* and the *knock-knee model* have been extensively studied. It is well known that the general *CR* and *SBR* problems under these two models are *NP*-complete.[1,2] Many heuristic algorithms for the Manhattan mode *CR* problems have been proposed.[3,4,5,6,7,8,9] Baker's *et al.* algorithm[3] is the only one known to have a provably good performance with respect to the channel width. There are no known algorithms to determine routability of the Manhattan mode *SBR* problem. The only

known Manhattan mode *SBR* algorithms are heuristic algorithms.[10] Several special Manhattan mode *CR* and *SBR* problems have been investigated. The *CR* and *SBR* problems are called *compatible*, if there is at most one terminal point along each grid line. The compatible *CR* can be solved efficiently under the Manhattan routing model using the algorithm developed by Hashimoto and Stevens.[11] Gonzalez and Zheng[12] developed an efficient algorithm for two-layer wiring of any two-terminal-net compatible *SBR* problem routable under the Manhattan model. The algorithm stretches the layout introducing a fixed number of tracks and columns.

Unlike Manhattan mode wire layouts, which can be wired in two layers, most knock-knee mode wire layouts require more than two layers; however, problem instances that cannot be routed under that model become routable under the knock-knee wiring model. The two-terminal net knock-knee mode *CR* problem can be solved efficiently by Frank's algorithm,[13] and improved algorithms have been presented in several papers.[14,15,16,17] Approximation algorithms for the multiterminal-net knock-knee mode *CR* problem have been investigated.[15,16,18] Gao and Kaufmann's algorithm[18] has the smallest approximation bound, and when incorporated with Weiners-Lummer's layer assignment strategy,[19] a three layer wiring can be generated. Knock-knee mode two-terminal net *SBR* algorithms had been developed.[13,20] The currently best knock-knee mode multiterminal-net *SBR* algorithms reduce the multiterminal net *SBR* problems into two-terminal net *SBR* problems by vertically and horizontally stretching the layout.[20,21] For multilayer wiring the fundamental result is an efficient algorithm to wire in four layers any non-overlapped wire layout.[22]

The non-overlap wiring models prohibit wire segments in different layers to overlap in parallel (i.e., in at least one unit segment). The reason for restricting wires from overlapping is the undesired capacitance introduced by the overlap of two wires. Current VLSI technology allows overlap of metal layers. As the ratio gates/chip increases, more layers are needed for the wiring. Therefore, wiring models that allow wire overlap are becoming more popular.

In the past few years, the *CR* and *SBR* problems under the wire overlap model received considerable attention. Several different wire overlap wiring models have been considered. These models differ in the restrictions on the length of wire overlaps, the number of layers in which wires can overlap, or the number of separation layers between any two overlapped wire segments. Most of the previous wire overlap mode routing algorithms are for the *CR* problem.[23,24,25,26,27,28,29,30,31] In contrast, the investigation on wire overlap mode *SBR* problem has been limited.[32]

In this paper we present a new algorithm for the two-overlap *SBR* problem. An instance of the *SBR* problem consists of a rectangle R and a set $N = \{n_1, n_2, \ldots, n_m\}$ of m two-terminal nets, each formed by two terminal points on the boundary of R. Every grid point on the boundary of R has at most one terminal point. The objective is to connect the two terminal points in each net by introducing a set of wires inside R in k layers that satisfy the constraints in the two-overlap wiring model. In this model there are at most two wire segments assigned to the same

track or column unit segment in all the layers. The best known two-overlap routing algorithms are given in Refs. 27, 32. The algorithm by Cong *et al.*[27] wires in four layers any multiterminal net *CR* problem using $\lceil d/2 \rceil + 2$ tracks. Kaufmann's algorithm[32] generates a nine-layer two-overlap wiring for any two-terminal net routable *SBR* problem. For multiterminal net *SBR* problems Kaufmann[32] developed an algorithm that generates nineteen-layer layouts. The wirings generated by Kaufmann's algorithms have the property that all the wire segments in each layer are either vertical or horizontal, but not both.

We present an algorithm that given any two-overlap routable *SBR* problem, adds at most two tracks and two columns, and wires it in seven layers. When the set of n terminal points is initially ordered, the time complexity for our algorithm is $O(n)$. The layouts generated by our algorithm have the property that all the wire segments in each layer are either vertical or horizontal, but not both. Most of the vias introduced by our algorithm join wires in adjacent layers. The rest of the vias join wires in adjacent horizontal layers, and are located along two columns. Our algorithm can be generalized to solve the multiterminal-net *SBR* problem by transforming each multiterminal net into several two-terminal nets and stretching the layout.[20,21]

2. Preliminaries

Rectangle R is partitioned by a uniform grid L with h tracks (horizontal lines) and w columns (vertical lines). The intersection of a track and a column, or a track or a column with R is referred to as a *grid point*. An *edge* is a horizontal or vertical line segment that joins two adjacent grid points, and a *grid edge* is an edge that does not overlap with R. Each grid point at the boundary of R (excluding the corners of R) contains at most one *terminal point*. Let $N = \{n_1, n_2, \ldots, n_m\}$ be a partition of the set of terminal points into two-element sets called *nets*. Figure 1 gives a problem instance with $h = w = 5$ and $m = 10$.

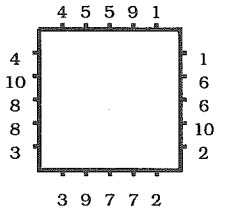


Fig. 1. A problem instance.

A k-layer wiring consists of k identical copies $L_1, L_2, \ldots, L_k$ of the grid L inside R, where L_i is above L_{i+1}. We refer to L_i as *layer i*. A k-layer layout for an *SBR* problem instance (denoted by $I = (R, N)$), is characterized by two mappings: wire

layout and layer assignment of the wire segments. A wire layout maps each net n_i in N to a wire W_i, which is a connected subgraph of L including both terminal points of net n_i. We refer to the set of wires $W = \{W_1, W_2, \ldots, W_m\}$ as *layout* of $I = (R, N)$. We say that W is an *l-layout* if at most l wire segments in W share any grid edge of L, and a *1-layout* is also called a *planar layout*. The layer assignment associates each wire segment on a grid edge e of L to its corresponding grid edge e' in a layer L_i, $1 \leq i \leq k$, in such a way that if two wire segments of wire W_i incident at a grid point p are assigned to layers i_1 and i_2 $(i_1 \leq i_2)$, then no other wire segment of a wire W_j $(i \neq j)$ incident at point p is assigned to a layer l, for all $i_1 \leq l \leq i_2$. When $i_1 < i_2$, we say that wire W_i has a *via* from layer i_1 to layer i_2 at point p. There could be several (disjoint) vias at the same grid point p when there are four or more layers, since more than one wire may be incident at grid point p. In what follows we refer to a horizontal (vertical) wire segment as an *hw*-segment (*vw*-segment).

A track or column is said to be *empty* if there are no terminal points at its intersection with R. Each track and column in a 2-layout consists of two *slots*, and wire segments may be assigned to either slot. The *corresponding track or column slot for a terminal point* are the track or column slots where the terminal point is located. The *corresponding track and/or column slots for a net* are the corresponding track and/or column slots for the terminal points of the net.

The bottom, top, left and right sides of R are labeled b, t, l and r, respectively. The set of nets N is partitioned into groups N_{rt}, N_{rb}, N_{lb}, N_{lt}, N_{tt}, N_{rr}, N_{bb}, N_{ll}, N_{tb}, and N_{lr}, where N_{xy}, for $x, y \in \{t, b, l, r\}$, represent the set of nets with one terminal located on side x of R and the other terminal located on side y of R. The net labeled i in Fig. 1 is in the i^{th} class defined above. A net is type *TB* if all its terminal points are located on the top and/or bottom sides of R (N_{tt}, N_{tb}, N_{bb}); it is type *LR* if all its terminal points are located on the left and/or right sides of R (N_{ll}, N_{lr}, N_{rr}); and it is type *neighbor* if its terminal points are located on two adjacent sides of R $(N_{rt}, N_{rb}, N_{lt}, N_{lb})$. Let $N_{TB} = N_{tt} \cup N_{tb} \cup N_{bb}$, $N_{LR} = N_{ll} \cup N_{lr} \cup N_{rr}$, and $N_{NN} = N_{rt} \cup N_{rb} \cup N_{lt} \cup N_{lb}$. A net is called a *trivial net* if both of its terminal points are located on the same track or column. The *natural bend number of a net* is the number of bends in a wire with least number of bends connecting the two points of the net. For example, the natural bend number of trivial nets is zero, for nets in N_{NN} is one and for non-trivial nets in N_{TB} is two. The *natural bend number of a problem instance is the sum of* the natural bend numbers of all its nets. The natural bend number of the problem instance in Fig. 1 is 16.

We say that net n_i *spans over the vertical (horizontal) line* l if net n_i contains a terminal point on each side of line l. The *density of line* l is the number of nets that span over line l. In Fig. 1, each vertical or horizontal non-grid line l that partitions R has density 3. The *vertical (horizontal) density*, $D_v(N)$ $(D_h(N))$ is defined as the maximum density of any vertical (horizontal) non-grid line. The problem instance given in Fig. 1 has $D_v(N) = 3$ and $D_h(N) = 3$. Obviously, if for

problem instance $I = (R, N)$ either $D_v(N) > 2h$ or $D_h(N) > 2w$, then there is no two-overlap wiring of the nets N in R. Therefore, necessary conditions for the existence of a wiring of N inside R are $D_v(N) \leq 2h$ and $D_h(N) \leq 2w$. It is not known whether or not these conditions are sufficient for the existence of a wiring of N in R. In what follows we "corrupt" our notation and say that a problem instance is *routable* if $D_v(N) \leq 2h$ and $D_h(N) \leq 2w$. Note that under this corrupted notation we might be calling some problem instances routable even though they are unroutable. It is interesting to note that any problem instance with $h = w$ is routable. This fact follows from Lemma 1 where we establish a relationship between h, w, D_v and D_h.

Lemma 1. *For problem instance* $I = (R, N)$,

(i) if $h \leq w$ *then* $D_h(N) \leq 2w$;
(ii) if $h \geq w$ *then* $D_v(N) \leq 2h$; *and*
(iii) if $h = w$ *then* $D_h(N) \leq 2w$ *and* $D_v(N) \leq 2h$.

Proof. Since the proof of the three parts is similar, we only prove (i). The only nets that can contribute to $D_h(N)$ are the nets in N_{tb}, N_{LR}, and N_{NN}. Each of these nets contributes at most one to $D_h(N)$. Therefore, $D_h(N) \leq |N_{tb}| + |N_{LR}| + |N_{NN}|$. Since each of these nets consist of two terminal points located on the boundary of R, there is at most one terminal point at each grid point in R (excluding the corners of R), and $h \leq w$, it must be that $2|N_{tb}| + 2|N_{LR}| + 2|N_{NN}| \leq 2h + 2w \leq 4w$. Therefore, $D_h(N) \leq 2w$. □

For problem instances that are not routable, one may introduce additional tracks or columns until it becomes routable. Hereafter we assume without loss of generality that $h \leq w$. Given a routable problem instance $I = (R, N)$, our algorithm extends the rectangle with at most two additional tracks and two columns, and constructs a 7-layer two-overlap wiring. Our procedure for constructing 2-layouts is similar to the one that constructs a planar layout for the *SBR* compatible net problem.[12] However, the layer assignment strategy in this paper is much more complex, and the underlying wiring models are quite different. We should also point out that the 2-layout constructed by our routing procedure, and our layer assignment strategy are different from the ones given in Ref. 32. The wires in the 2-layouts constructed by the algorithm in Ref. 32 do not have a minimum number of bends. In our 2-layouts at least two-thirds of the nets are connected by wires with a number of bends equal to the natural bend number of the net. The remaining nets (at most one-third) are connected by wires with a number of bends that exceeds by two the natural bend number of the net. There are problem instances (Fig. 2) for which there does not exist a 2-layout in which more than 66% of the nets are connected by wires with a number of bends equal to the natural bend number of the net. We should point out that constructing a 2-layout is simple, the difficulty arises in constructing a 2-layout that is seven-layer wirable. It is worth noting that four layers are normally

required for wiring a planar layout.[22] One would expect that for 2-layouts eight layers are required. In general this might be true, but for the specific 2-layouts that we construct, only seven layers are required.

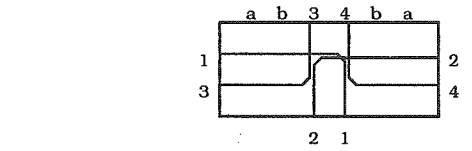
3. Routing Algorithm

We present procedure *ROUTE* that for any two-terminal-net two-overlap routable *SBR* problem instance constructs a 2-layout. Because of some special properties the 2-layouts are wirable in only seven layers (Sec. 4). Procedure *ROUTE* constructs a 2-layout by executing the following procedures: extension (*EXT*), trivial net routing (*ROUTE-TN*), neighbor net routing (*ROUTE-NN*), routing *LR* nets and a subset of *TB* nets (*ROUTE-LRTB* and *ROUTE-LR*), and routing the remaining problem (*ROUTE-REM*). First we outline the basic steps of our procedures and justify some of their critical steps. Then we formally define our procedures.

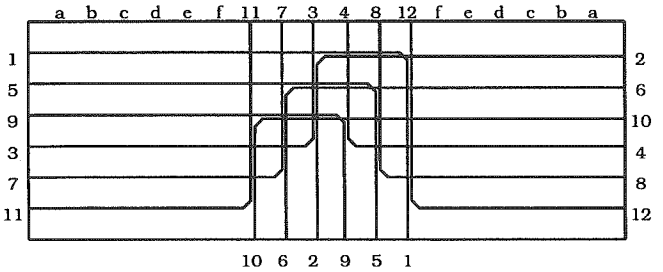
The problem instance is given by $I = (R, N)$, where R is a rectangle with h (w) interior tracks (columns) and N is the set of m two-terminal nets. Remember that we assume instance I is routable and $h \leq w$. In procedure *EXT*, one additional empty track is introduced next to the top boundary. The slots in this track are labeled t and t'. All of the trivial nets are routed (*ROUTE-TN*) in the obvious way in their corresponding track or column slots. In the neighbor net routing step (*ROUTE-NN*), each neighbor net is connected by an "L" shaped wire in its corresponding track and column slots.

Procedure *ROUTE-LRTB* routes a subset of nets that includes either all *TB*, or all *LR* nets. At each iteration in *ROUTE-LRTB*, two unrouted nets are selected, a *TB* net and an *LR* net. Then the two nets are routed in their corresponding track and column slots, and in certain cases in the track slot labeled t. Procedure *ROUTE-LRTB* terminates when either all *LR* or all *TB* nets have been routed. Rectangle R is vertically stretched by introducing two new grid lines without terminal points nor vertical wires, and in some cases R is also stretched horizontally. In procedure *ROUTE-LR*, each unrouted *LR* net (if any) is routed in a distinct column slot and in its corresponding track slots.

At this point one may expect that the routing of the remaining nets is straightforward. Unfortunately, this is not the case. Consider the problem instance in Fig. 2(a). After routing nets 1 through 4, it is impossible to wire both nets a and b. However, the problem instance can be wired by re-routing a net and introducing two additional bends (see the following procedure). By introducing additional nets, as suggested by Fig. 2(b), we can generate problem instances in which at most 66% of the nets are connected by wires with a number of bends equal to the natural bend number. Our algorithm must backtrack before a 2-layout may be generated for all possible problem instances. The reason why we do not backtrack to the beginning of the algorithm is that the resulting problem at this point is in general significantly



(a)



(b)

Fig. 2. Problem instances for which we must backtrack.

simpler than the original problem. The following procedure removes a set of wires, and then generates a 2-layout.

If all nets have been routed, then procedure *ROUTE* terminates; otherwise, we execute procedure *ROUTE-REM*. The procedure begins by removing wire segments inside a rectangle that includes all tracks in R, and a set of adjacent columns in R. The resulting problem is referred to by $I' = (R', N')$. The set of nets N' is partitioned into *runs*, such that the vertical density of each *run* is one, and the total number of *runs* is equal to the vertical density of the unrouted nets. At each iteration we route all nets in one or two *runs*. The 2-layout for this new problem plus the previously introduced wire segments (that were not deleted at the beginning of this procedure) form the 2-layout which is the output of procedure *ROUTE*.

Let us explain in detail algorithm *ROUTE* when invoked with problem instance $I = (R, N)$. We use the example given in Fig. 3 to illustrate our algorithm. By convention $h \leq w$ and instance I is routable, i.e., $D_h(N) \leq 2w$ and $D_v(N) \leq 2h$. In Subsection 3.1, we introduce our notation, and in Subsection 3.2 we present our procedure for routing trivial and neighbor nets. Procedures *ROUTE-LRTB* and *ROUTE-LR* are explained in Subsection 3.3, and Subsection 3.4 explains procedure *ROUTE-REM*. Finally, in Subsection 3.5 we establish our main result for this Section.

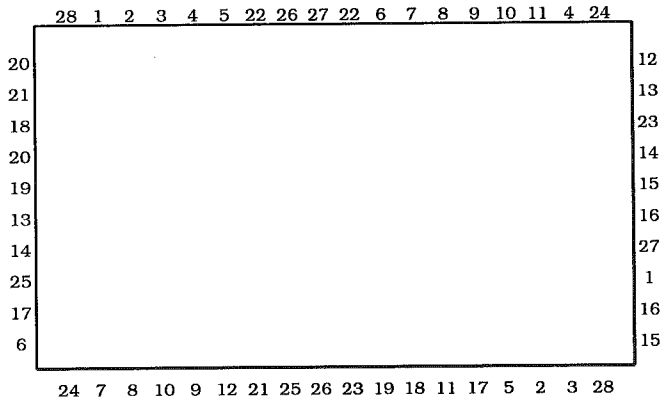


Fig. 3. Problem instance $I = (R, N)$.

3.1. Notation

Procedure *EXT* introduces an empty track between the top side of R and the topmost track. For convenience we shall not modify the value of h, so actually, after this iteration the number of tracks is not h, but $h + 1$. The track slots in this new empty track are labeled t and t'. All track and column slots are labeled *available* for routing, except for track slot t which is labeled *reserved*, and track slot t' which is labeled *unavailable* for routing. We uniquely *match* each terminal point from an unrouted net with one of its corresponding track or column slots that is *available*. For trivial nets, only one of its terminal points is *matched* with a track or column slot. From the initial conditions we know that a matching is always possible. Hereafter, an *available* track or column slot is said to be either *matched* with a terminal point from an unrouted net, or *not-matched*. Once a set of nets is routed in an *available* track or column slot, the slot is relabeled *unavailable* or *reserved*.

Let *mats* (*macs*) represent the number of *matched available* track (column) slots, and let *nmats* (*nmacs*) the number of *not-matched available* track (column) slots. A trivial net whose terminal points are located on the top and bottom side of R is called a *c-trivial* net. Let T_{TB} be the set of c-trivial nets, let $u(T_{TB})$ be the set of unrouted c-trivial nets, and let $|u(T_{TB})|$ denote the number of unrouted c-trivial nets. In Lemma 2 we establish that some important properties hold just after labeling the tracks. These invariants are needed to establish correctness of our algorithm. For example, inequality (i) below will be used to show that there is enough space to route the remaining set of unrouted nets.

Lemma 2. *Just after the track and column slots are labeled, the following statements hold.*

(i) $mats + |u(T_{TB})| \leq macs + 2nmacs$.

(ii) *Each terminal point from the unrouted LR (TB) nets is uniquely matched with one of its corresponding available track (column) slots, except for terminal points from trivial nets, in which case, only one of the terminal points is matched.*

(iii) *Track slot t does not contain wires.*

(iv) *Each non-grid vertical line intersects the hw-segments from each net at most once.*

Proof. The proof for (ii) is straightforward, and (iii) and (iv) hold because there are no wires. Let us now prove (i). Since each neighbor and c-trivial net is *matched* with one column slot, each non-trivial TB net is *matched* with two column slots, and $T_{TB} \subseteq N_{TB}$, we know that $macs = |N_{NN}| + 2|N_{TB}| - |T_{TB}|$. By definition, $nmacs = 2w - macs$. Combining the two expressions we know that $macs + 2nmacs = 4w - |N_{NN}| - 2|N_{TB}| + |T_{TB}|$. Since each neighbor net has one terminal point on the top or bottom side of R, each TB net has two terminals on the top and bottom side of R, and the total number of terminal points located on the top and bottom side of R is at most $2w$, we know that $|N_{NN}| + 2|N_{TB}| \leq 2w$. Substituting in the above expression we know that $macs + 2nmacs \geq 2w + |T_{TB}|$. Since $w \geq h$ and $2h \geq mats$, we know that $macs + 2nmacs \geq mats + |T_{TB}|$.

Just after the track and column slots are labeled, $|T_{TB}|$ is equal to $|u(T_{TB})|$. Substituting in the above inequality, we obtain (i). Therefore, (i)–(iv) hold just after the track and column slots are labeled. □

3.2. Trivial and Neighbor Net Routing

Each trivial net is routed by *ROUTE-TN* in the obvious way in the track or column slot *matched* with one of its terminal points. Each track or column slot involved in the routing of the trivial net is labeled *unavailable*. Therefore, in each iteration of *ROUTE-TN* either one net in $u(T_{TB})$ is deleted or *mats* decreases by one, and *macs* decreases by at most one. The value of $u(T_{TB})$ is zero when procedure *ROUTE-TN* terminates. We claim that (i)–(iv) in Lemma 3 hold when procedure *ROUTE-TN* terminates.

Lemma 3. *Statement (i)–(iv) in Lemma 2 hold when procedure ROUTE-TN terminates.*

Proof. The proof is omitted since it is straightforward. □

Next we apply procedure *ROUTE-NN*, to route the set of neighbor nets. At each iteration we select a neighbor net, and route it by an "L" shaped wire. Then, we label the track and column slots *matched* with the net's terminal points *unavailable*. Therefore, just after each iteration *mats* and *macs* decrease by one. Figure 4 shows

our example after procedures *EXT*, *ROUTE-TN*, and *ROUTE-NN* are executed. Note that in this example there are no trivial nets.

Lemma 4. *Statement (i)–(iv) in Lemma 2 hold just after procedure ROUTE-NN terminates.*

Proof. The proof is omitted since it is straightforward. □

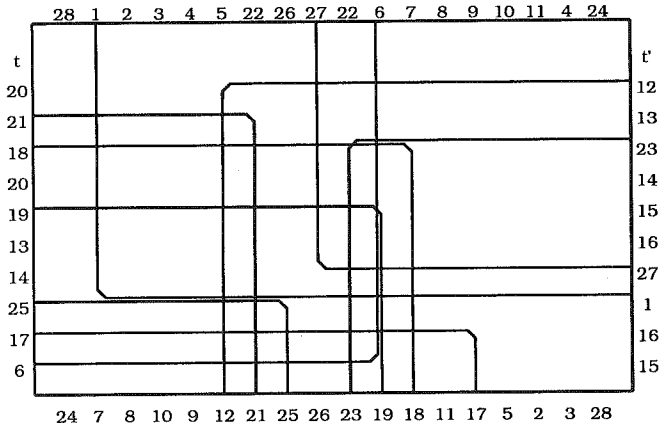


Fig. 4. Our example after introducing the new track and routing all neighbor nets.

3.3. Procedures ROUTE-LRTB and ROUTE-LR

We apply procedure *ROUTE-LRTB* which iterates until either all *LR* or all *TB* nets have been routed. Then, procedure *ROUTE-LR* routes all the unrouted *LR* nets (if any). Let us formally define each of these procedures.

Procedure *ROUTE-LRTB* iterates until all *LR* or all *TB* nets have been routed. At the beginning of the first iteration we define a window G inside R, and at the end of each iteration G is updated. Window G is the smallest rectangle inside R such that all terminal points from the unrouted *LR* (*TB*) nets can be horizontally (vertically) projected to its boundary. Project terminal points from unrouted *LR* and *TB* nets to their nearest points in G. We say that a corner of G is *doubly-covered* if two terminal points were projected to it. Let *left (right)* represent the x-coordinate value of the left (right) boundary of G.

Two unrouted nets (one from *LR* and one from *TB*) are routed at a time. The specific nets to be routed at each iteration depend on whether there are *doubly-covered* corners in G.

Case 1: There is a *doubly-covered* corner in G.

Assume that the *doubly-covered* corner is located on the bottom-right corner of G and that t is above G; the other cases are treated similarly. If the rightmost *available* column is on the right boundary of the rectangle G, then Fig. 5 shows the actions performed in this case. Our "visual" notation is defined as follows. A dotted line represents a track or column slot that is marked *unavailable* at this step; the dashed line represents the *reserved* track slot t; and a solid thick line represents a net routed at this step. In this case a knock-knee is introduced. Figure 5 does not show all the wires introduced for the two nets (the missing part of the wires only occupy part of the dotted track or column slot up to where it ends). On the other hand, if the rightmost *available* column is to the right of rectangle G, then the LR net is routed in that column. As a result of this, that column is labeled *unavailable*, and the column on the right boundary of G is labeled *not-matched available*.

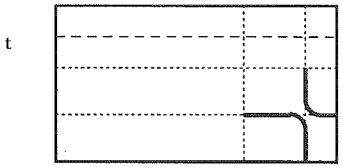


Fig. 5. Wires introduced for Case 1.

Case 2: There are no *doubly-covered* corners in G.

The two Subcases 2(a) and 2(b) that arise are shown in Figs. 6(a) and (b), respectively. The line segments intersecting the boundary of R show the locations where the terminal points from unrouted nets were projected to the corners of G.

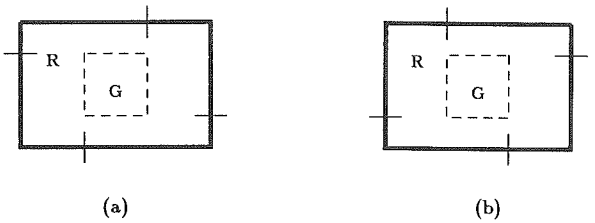


Fig. 6. There is no doubly-covered corner in G.

Assume without loss of generality that t is above G. If Case (a) ((b)) applies and the rightmost (leftmost) *available* column is on the right (left) boundary of G, then the wire segments introduced in this case are shown in Fig. 7(a) ((b)) and corresponds to the case shown in Fig. 6(a) ((b)). After the iteration the dotted-dashed line represents the new *reserved* track slot t, and the previous track slot t becomes *unavailable*. On the other hand, if Case (a) ((b)) applies and the rightmost

(leftmost) *available* column is to the right (left) of rectangle G, then the LR net is routed in that column. As a result of this, that column is labeled *unavailable*, and the column on the right (left) boundary of G is labeled *not-matched available*.

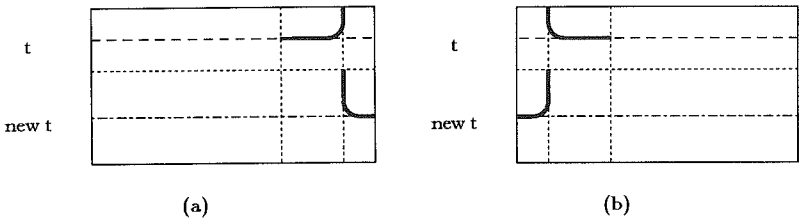


Fig. 7. Wires introduced in Case 2 when t is above G.

Clearly, at each iteration *mats* and *macs* decrease by exactly two. At the end of the iteration, the track labeled t is not *matched* with a terminal point from an unrouted net, and track t does not have a wire from column *left* to column *right* (*left* and *right* are defined with respect to the G at the end of the iteration). The above process is repeated as long as there are unrouted LR and TB nets. Procedure *ROUTE-LRTB* terminates when there remain no unrouted TB or LR nets.

Figure 8 shows the wires introduced in our example by procedure *ROUTE-LRTB*, but for clarity does not show the wires introduced during the previous steps. In all our figures we use a solid dot on the boundary of R to indicate the location of an *unavailable* track or column slot (in this case the track or column slot was *unavailable* when procedure *ROUTE-LRTB* began). Similarly, an X represents a *not-matched available* track or column slot. In our example, procedure *ROUTE-LRTB* performs the following steps:

1. Rectangle G is defined. Nets 28 and 15 are wired from the bottom-right corner of G (Case 1), and G is updated.
2. Nets 24 and 16 are wired (Case 2(a)). Track slot t is now the track slot *matched* with the bottommost terminal point from net 16, and G is updated.
3. Nets 7 and 14 are wired from the bottom-left corner of G (Case 1), and G is updated.
4. Nets 2 and 20 are wired from the top-left corner of G (Case 1), and G is updated.
5. Nets 8 and 13 are wired from the bottom-left corner of G (Case 1), and G is updated.

In Lemma 5, we establish that five statements (including statements (i)–(iv) of Lemma 4) hold at the beginning of each iteration of procedure *ROUTE-LRTB*, and when procedure *ROUTE-LRTB* terminates.

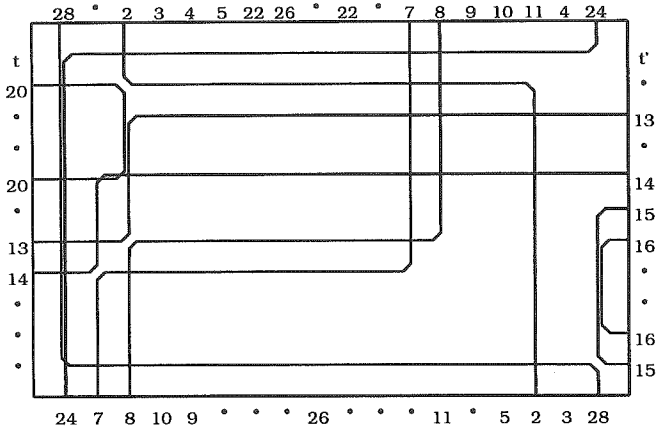


Fig. 8. Wires introduced by procedure *ROUTE-LRTB*.

Lemma 5. *Statements (i)–(v) hold at the beginning of each iteration of ROUTE-LRTB, and when procedure ROUTE-LRTB terminates.*

- *(i)* $mats \leq macs + 2nmacs$.
- *(ii) Each terminal point from unrouted LR (TB) nets is uniquely matched with one of its corresponding available track (column) slot.*
- *(iii) Track slot t is not in a track inside G and it does not contain hw-segments from column left to column right.*
- *(iv) Each vertical line that intersects the open interval from column left to column right intersects the hw-segments from each net at most once.*
- *(v) All the terminal points from unrouted LR (TB) nets can be horizontally (vertically) projected to the boundary of G.*

Proof. By Lemma 4 and the definition of G, we know that statements (i)–(v) hold at the beginning of procedure *ROUTE-LRTB*. It is simple to show that if (i)–(v) hold at the beginning of an iteration, they also hold at the end of the iteration. Thus, the lemma follows by induction. □

Figure 9 shows all the wires introduced in our example by procedure *ROUTE-LRTB*. If all nets have been routed, then let *left* (*right*) be the left (right) boundary of last rectangle G defined in *ROUTE-LRTB*; unless no rectangle G was ever defined, in which case *left* and *right* are adjacent columns in the middle of rectangle R. If there remain only *TB* unrouted nets, then let *left* (*right*) denote the leftmost (rightmost) column where a terminal point from an unrouted *TB* net is located, unless such a column contains the *vw*-segment of an *LR* net, in which case, *left* (*right*) is set to the column immediately to its right (left). On the other hand, if there remain only *LR* unrouted nets, let *left* (*right*) be the leftmost (rightmost)

column that is *available*. Rectangle R is stretched by introducing two vertical grid lines (without terminal points nor vertical wires) between columns *left-1* (*right*) and *left* (*right+1*). The horizontal wires intersecting these columns are stretched and preserve the previous connectivity. The values of *left* and *right* are updated to point to these new columns. Rectangle R' includes all tracks, and columns $left, left+1, \ldots, right$ in R. In Sec. 4 it will be evident why we have introduced the two additional columns. If all the nets have been routed then we proceed to the layer assignment phase (Sec. 4). On the other hand, if all the LR nets have been routed, we apply procedure *ROUTE-REM*; otherwise we apply procedure *ROUTE-LR*. These procedures are defined below.

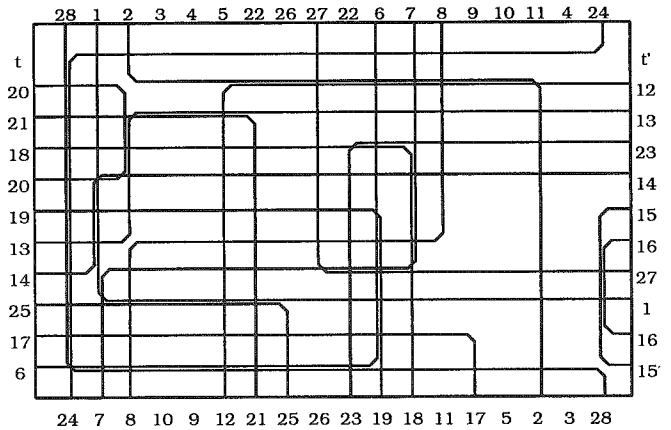


Fig. 9. All wires introduced after *ROUTE-LR*.

In procedure *ROUTE-LR* we route all the remaining unrouted LR nets. In our example, all LR nets have been routed (see Fig. 8), so procedure *ROUTE-LR* does not introduce new wires. Let us now consider the case when there remain unrouted LR nets. Each unrouted LR net is routed in the two track slots *matched* with its terminal points, and in a *not-matched available* column slot. An N_{ll} (N_{lr} or N_{rr}) net is routed in the leftmost (rightmost) *not-matched available* column slot. By Lemma 5(i) we know that at least one such column slot exists for each of the remaining unrouted LR nets. The track and column slots involved in the routing of each of these nets are labeled *unavailable*. At the end of procedure *ROUTE-LR*, rectangle G is redefined as a single grid point inside the previous G. In Lemma 6 we establish some important properties of the wiring generated by procedure *ROUTE-LR*.

Lemma 6. *Statements (i)-(v) hold at the beginning of procedure ROUTE-LR, and just after procedure ROUTE-LR terminates.*

Proof. By Lemma 5 it follows that (i)–(v) hold at the beginning of procedure *ROUTE-LR*. From the above discussion, we know that if (i)–(v) hold at the beginning of procedure *ROUTE-LR*, then they also hold when procedure *ROUTE-LR* terminates. □

3.4. Procedure ROUTE-REM

As mentioned before, we must backtrack in order to be able to route the remaining nets. Let us now define the remaining problem, which is considerably simpler to route than the original problem. From the previous procedures we know that the only unrouted nets are *TB* nets. The remaining problem, which we shall refer to as $I' = (R', N')$, is defined by deleting some of the previously introduced wire segments. Specifically, all the *hw*-segments segments located on a track from column *left* to column *right*, and all the *vw*-segments from column *left*+1 to column *right*-1 are deleted. In our example, columns *left*+1 and *right*-1 contain terminal points of routed nets. In general, this may not be true. We claim that at most one terminal point in this column could belong to an unrouted *TB* net. In this case, we introduce a new horizontal track at the bottom of the rectangle, and t' will be one of the track slots in the bottom track. The other slot in the bottom track, and the old track slot t', are used for wire segments that connect the unrouted terminal points to the boundary of R', without introducing any segments inside R'. Figure 10 shows the remaining wires after applying this step to the 2-layout given in Fig. 9.

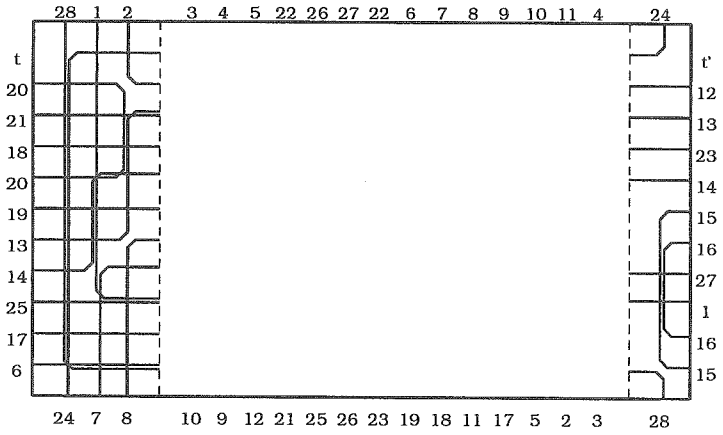


Fig. 10. Remaining wires.

It is important to remember that rectangle R' includes all tracks in R, and the columns $left$, $left + 1, \ldots, right$. The terminal points in R' consist of all the terminal points from previously unrouted nets plus new terminal points located at the intersection of the boundary of R' with a deleted wire (these points of intersection

are labeled with the name of the corresponding net). For R', we have a set N' of two terminal nets. Figure 11 shows the rectangle R' for our example. Note that there may be more than one terminal point at the intersection of a track with the left or right boundary of R'. However, if this is the case, at least one of them belongs to a trivial net, i.e., the corresponding point on the opposite side of R' has the same label, and either the track has two trivial nets, or one trivial net and at most another terminal point that belongs to an unrouted net in N'. It is simple to show that the set N' of nets can be partitioned into N'_{TB}, N'_{NN}, and trivial N'_{LR} nets. This simplifies considerably the routing problem, and is the main reason why one should not eliminate the first few steps in our procedure. Note that each of the nets of N' may not be of the same type as in N. Figure 11 shows the problem $I' = (R', N')$ generated for our example, and Lemma 7 establishes that I' is routable.

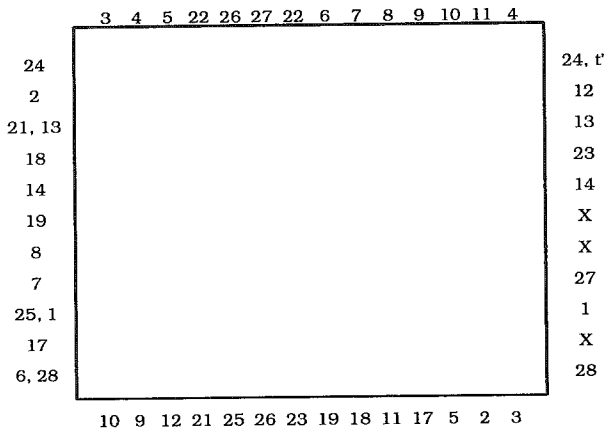


Fig. 11. Problem instance I' for our example.

Lemma 7. *Problem instance I' constructed at the beginning of procedure ROUTE-REM is routable, i.e. the horizontal and vertical densities of problem instance I' satisfy the necessary conditions for routability.*

Proof. The proof follows from Lemma 6, and the discussion in the previous paragraph. □

The final layout consists of the remaining wires in R (those not deleted at the beginning of this step) plus the wires introduced inside R' by *ROUTE-REM*. The remaining problem consists of introducing a set of wires inside R' to connect the nets in N'. This is a restricted version of the original problem, with the possible exception that there could be two terminal points on a track at the intersection

with the left or right boundary of R'. However, at least one of those two terminal points belongs to a trivial net in N'_{LR}. All the column and track slots are labeled *available*, except for track slot t' which is labeled *reserved*. Each terminal point from an unrouted net (except for trivial N'_{LR} nets in which case only one of its terminal points) is uniquely *matched* to one of its corresponding track or column slots. It is simple to show that a matching is always possible. Hereafter, an *available* track or column slot is said to be either *matched* with a terminal point from an unrouted net or *not-matched*.

Procedure *ROUTE-REM* begins by routing the trivial N'_{LR} and the trivial N'_{TB} nets. Each of these nets is routed in the obvious way in the *available* track or column slot *matched* with one of its terminal points, and such a slot is relabeled *unavailable*. Figure 12 shows our example after routing trivial nets. In our example there are no trivial N'_{TB} nets.

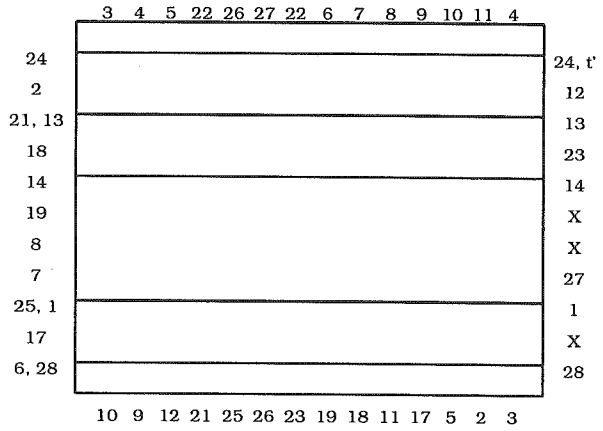


Fig. 12. Wiring trivial nets.

Note that each time a trivial N'_{LR} (N'_{TB}) net is routed, the vertical (horizontal) density of the unrouted nets, and the number of *matched available* track (columns) slots, decreases by one. Therefore, if problem instance I' was routable, then the resulting problem is also routable. The resulting problem for our example is shown in Fig. 13. Remember that a solid dot indicates the location of an *unavailable* track or column slot, and an X indicates the location of a *not-matched available* track or column slot.

Procedure *ROUTE-REM* groups the unrouted nets into sets called *runs*, which are constructed as follows. Initially, all unrouted nets are marked as *unvisited*. The nets in the first *run* are identified first, then the ones in the second *run*, and so forth. We identify the nets in the i^{th} *run* as follows. The first net in the i^{th} *run* is an unvisited net whose leftmost terminal point has the smallest x-coordinate value

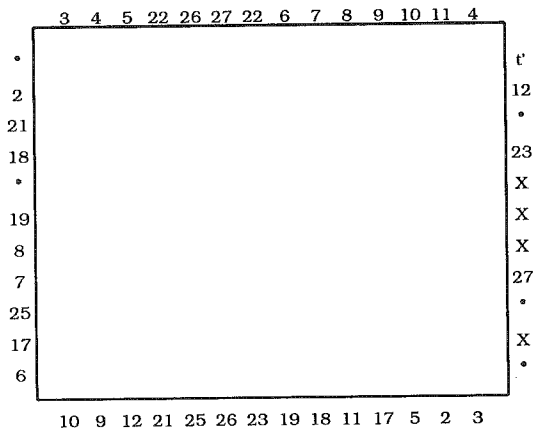


Fig. 13. Resulting problem after wiring trivial nets.

(in case of ties select any one of them). For $k = 2, 3, \ldots$, the k^{th} net in the i^{th} *run* is an unvisited net whose leftmost terminal has the smallest x-coordinate value, and such value is greater than or equal to the x-coordinate value of the rightmost terminal of the $(k-1)$ net in the i^{th} *run*. Once all the nets in the i^{th} *run* have been identified, we mark them as visited and proceed to find the $(i+1)$st *run*. This process is repeated until all unrouted nets are marked visited. Lemma 8 establishes an important property of *runs*.

Lemma 8. *The number of runs identified by procedure ROUTE-REM is less than or equal to the number of available track slots.*

Proof. The proof follows from Lemma 7, the fact that each *run* has density one, and the fact that the vertical density of the *runs* is equal to the number of *runs*. □

Depending on the number of neighbor nets in a *run*, the *run* is called a 0_n *run*, a 1_n *run* or a 2_n *run*. In our example, procedure *ROUTE-REM* identifies the *runs* given in Table 1. Since our example is small, most *runs* consist of only one net; however, in general, *runs* may contain several nets.

Procedure *ROUTE-REM* then applies the following three rules (in any order) until all *runs* are routed, or the rules do not apply. Later on we show how to route the remaining *runs* (if any).

(a) An unrouted 1_n *run* is routed in the track slot *matched* with its neighbor net, and in the column slots *matched* with the nets in the *run*. These column and track slots are labeled *unavailable*.

Table 1. Runs for our example.

Type	Runs
0_n	3, 4, 5, 9, 10
1_n	2, 6, 7, 8, 12, 17, 18-11, 19
2_n	21-22-23, 25-26-27

(b) If the two track slots *matched* with the neighbor nets in an unrouted 2_n *run* belong to the same track, then route the 2_n *run* in one of these track slots and in the column slots *matched* with the nets in the *run*. Label all of these column and track slots *unavailable*. The unused track slot *matched* with the neighbor nets in the *run* becomes a *not-matched* (but, still *available*) track slot.

(c) When there is a *not-matched available* track slot, and an unrouted 0_n *run*, route an unrouted 0_n *run* in the *not-matched available* track slot, and in the column slots *matched* with the nets in the *run*. Label all of these column and track slots *unavailable*.

The wiring generated by applying the above three rules to our example is given in Fig. 14. Note that rule (b) is not applied in this example.

After procedure *ROUTE-REM* applies these three rules as many times as possible, the only remaining unrouted *runs* (if any) are 0_n and 2_n *runs*. The unrouted nets in our example are given in Fig. 15. We claim that the remaining unrouted *runs* satisfy the conditions of Lemma 9.

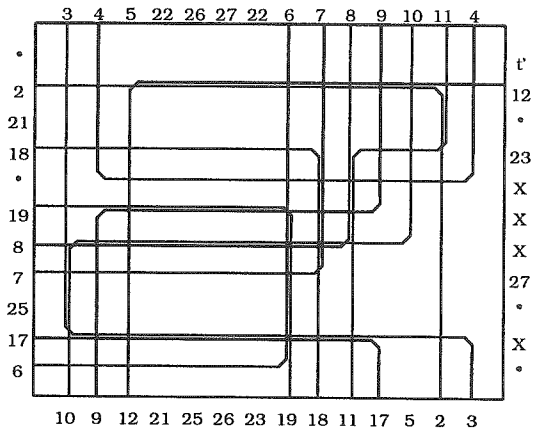


Fig. 14. Wiring of the *runs*.

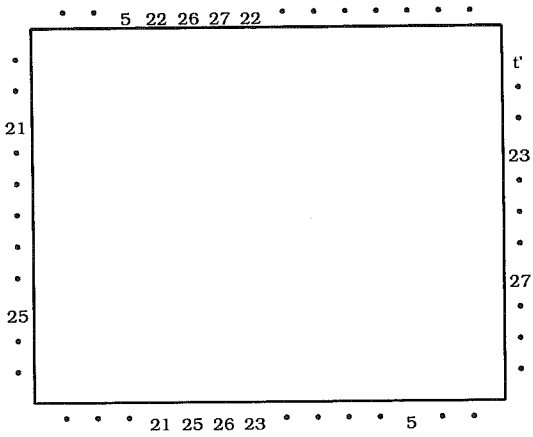


Fig. 15. Wiring of the *runs*.

Lemma 9.

1. *After every application of rules (a)–(c), the number of unrouted runs is less than or equal to the number of available track slots.*
2. *After procedure ROUTE-REM applies rules (a)–(c) as many times as possible, the number of unrouted* 0_n *runs is less than or equal to the number of unrouted* 2_n *runs.*

Proof. The proof follows from Lemma 8 and the fact that each time we apply rules (a)–(c), the vertical density of the unrouted nets, the number of *runs*, and the number of *available* track slots decreases by one. □

Now procedure *ROUTE-REM* routes a subset of the unrouted 2_n *runs*, and all the unrouted 0_n *runs*. The procedure iterates as long as there are unrouted 0_n *runs*. At the beginning of each iteration G' is defined as the smallest rectangle inside R' such that all the terminal points located on the left and right boundary of R' from the neighbor nets in the unrouted 2_n *runs* can be horizontally projected to it; and all the terminal points from the nets in the unrouted 0_n *runs* can be vertically projected to it. Now project each of these terminal points to their nearest point in G'. A corner in G' is called a *doubly-covered* corner of G' if two of the previously identified terminal points were projected to that corner. Procedure *ROUTE-REM* routes all the nets in a 0_n and in a 2_n *run*. The specific *runs* routed depend on whether there is a *doubly-covered* corner in G'.

Case 1: There is a *doubly-covered* corner in G'.

Assume without loss of generality that the top-left corner of G' is a doubly-covered corner, since the other cases can be treated similarly. The 0_n and 2_n *run*

containing the terminal points projected to the top-left corner of G' are routed at this iteration. Figure 16(a) shows the case when no knock-knees are introduced in track p; Figure 16(b) shows the case when our procedure does not introduce a three-bend wire, but introduces a knock-knee in track p; and Fig. 16(c) shows the case when our procedure introduces a three-bend wire and a knock-knee in track p. The routing is in the track slots *matched* to the terminal points of the two neighbor nets in the 2_n *run*. The columns slots involved in the routing of these nets are those that are *matched* to the terminal points of the nets in the two *runs*. All of these slots are labeled *unavailable* at the end of the iteration.

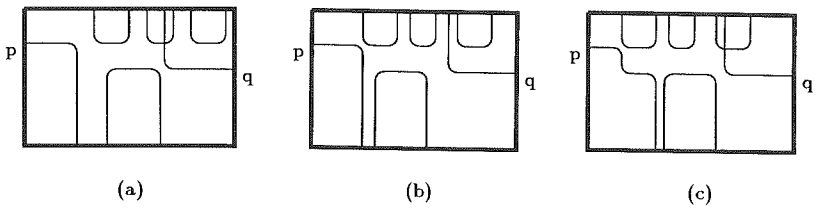


Fig. 16. Wiring for Case 1.

Case 2: There are no doubly-covered corners in G'.

The two cases are depicted in Fig. 17. The line segments intersecting the boundary of R indicate the terminal points from the previously identified nets projected to corners of G'. Assume without loss of generality that track t' is located above G'.

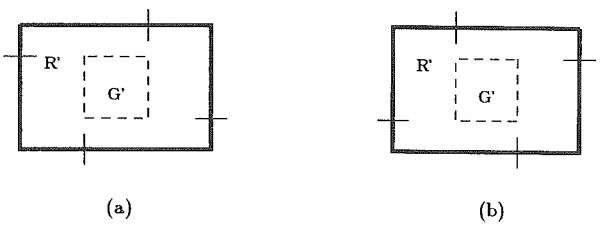


Fig. 17. There is no doubly-covered corner in G'.

If the case given in Fig. 17(a) ((b)) holds, we route the 2_n *run* that includes the net whose terminal point was projected to the top-left (top-right) corner of G', and the 0_n *run* that includes the net whose terminal point was projected to the top-right (top-left) corner of G'. In this case, track slot t' is used for routing, but after this step, label t' is assigned to track slot q. The specific routing is shown in Figs. 18(a)–(b). Here we only show the case when a neighbor net is connected by a three-bend wire.

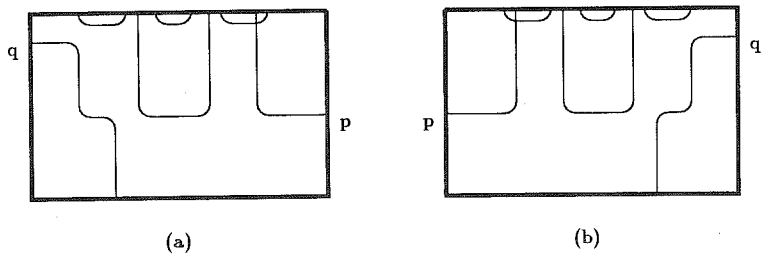


Fig. 18. Wiring for Case 2.

The above process is repeated until all the 0_n *runs* have been routed. In our example the *runs* 21-22-23, and 5 are routed as in Case 1. Figure 19 shows the wiring of the above two *runs*. If all the nets have been routed, then procedure *ROUTE-REM* and *ROUTE* terminate. Otherwise, each of the remaining 2_n *runs* is routed in the track slots *matched* to the neighbor nets in the *run*, and in the column slots *matched* to the nets in the *run*. All of these column and track slots are labeled *unavailable*. For our example, the remaining *run* 25-26-27 is routed using the track slots *matched* with the nets 25 and 27, the neighbor nets in the *run*. Figure 20 shows the wiring of the *run* 25-26-27. Procedures *ROUTE-REM* and *ROUTE* terminate

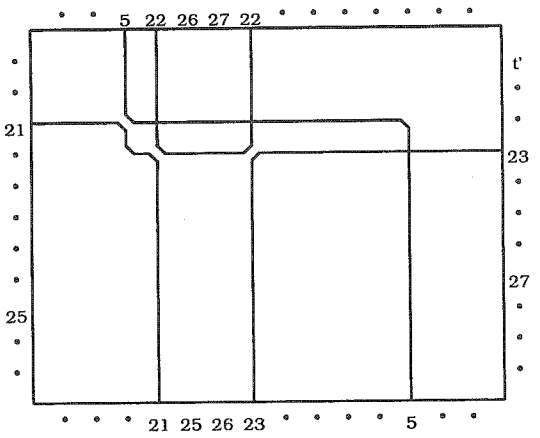


Fig. 19. Wiring of the runs after routing all 0_n runs.

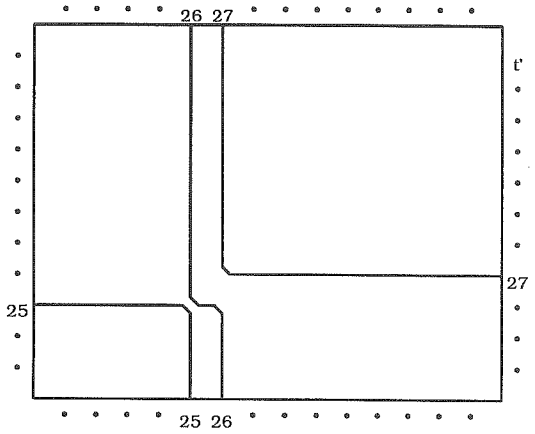


Fig. 20. Wiring of the remaining 2_n *runs*.

when all *runs* are routed. In Lemma 10 we establish that *ROUTE-REM* generates a 2-layout for I' and in Theorem 1, we establish that procedure *ROUTE* constructs a 2-layout for I.

Lemma 10. *Procedure ROUTE-REM routes problem instance I'.*

Proof. The proof follows from Lemmas 7–9 and the fact that the last two loops in *ROUTE-REM* route the remaining 0_n and 2_n *runs*. □

3.5. Summary

Theorem 1. *Procedure ROUTE constructs a 2-layout for any two-overlap routable instance $I = (R, N)$ of the SBR problem by adding at most two tracks and two columns in R. The time complexity of procedure ROUTE is $O(n)$, when the set of n terminal points is initially ordered.*

Proof. It is simple to show that all steps in procedure *ROUTE* can be implemented in $O(n)$ time, if the set of terminal points is initially ordered. The remaining part of the proof follows from Lemmas 2–6 and Lemma 10. □

Figures 21 and 22 illustrate the final wiring of the problems I' and I of our example.

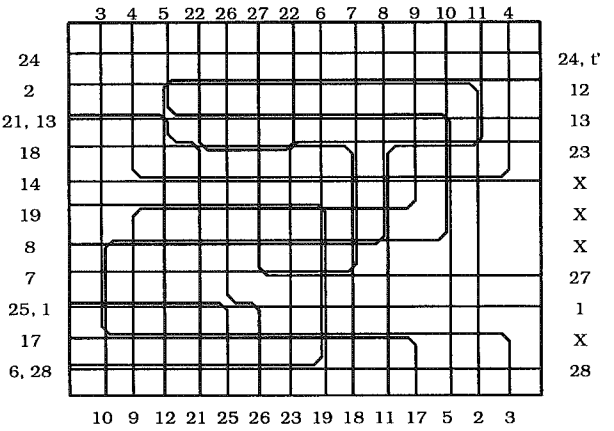


Fig. 21. Final wiring for the problem I'.

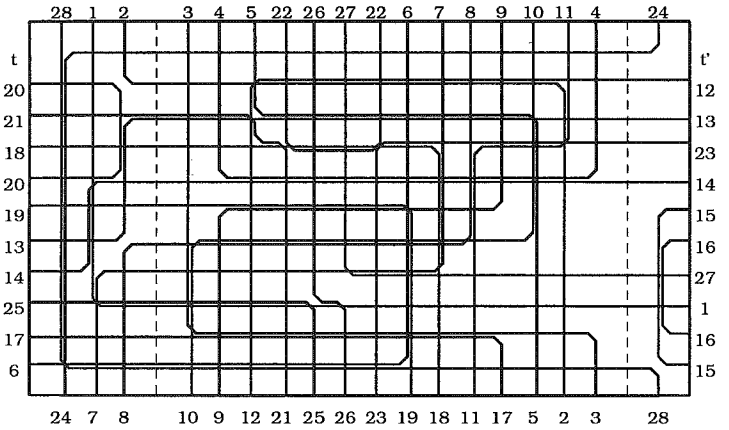


Fig. 22. Final wiring for the problem I.

4. Layer Assignment

The layer assignment phase comprises of assigning each wire segment to a layer in such a way that at each grid point no two wire segments and/or vias from different nets share a grid point in the same layer. The seven layers available for wiring are arranged from top to bottom in VHVHVHV order, where a layer is denoted V (H) if it has *vw*-segments (*hw*-segments) only. The three layers for the *hw*-segments are labeled *a*, *b*, and *c*; the layers for the *vw*-segments are labeled 1, 2, 3, and 4; and the layers are arranged from top to bottom, in the order 1, *a*, 2, *b*, 3, *c*, and 4. The

layer assignment of the wire segments introduced by procedures *ROUTE-LRTB* and *ROUTE-LR* inside R' is explained in Subsection (4.1). In Subsection (4.2) we explain our procedure for layer assignment of the wire segments introduced by procedure *ROUTE-REM* for the problem instance $I' = (R', N')$, and in Subsection (4.3) we explain the layer assignment for the wire segments introduced by procedures *ROUTE-LRTB* outside R'. For any particular problem instance, we perform the layer assignment of Subsections (4.1) and (4.3), or (4.2) and (4.3).

4.1. Layer Assignment for Wires Introduced by ROUTE-LRTB and ROUTE-LR Inside R'

Let us now consider the layer assignment of the wires, W', introduced by the procedures *ROUTE-LRTB* and *ROUTE-LR* inside R'. If the first (last) column has a *vw*-segment of an *LR* net which was introduced by *ROUTE-LRTB*, then we ignore that column when performing the layer assignment in this Subsection. Once the layer assignment of the remaining wires in R' is performed, the layer assignment for these columns is straightforward. This process greatly simplifies our notation. We begin by establishing in Lemma 11 some important properties of the wires introduced by procedures *ROUTE-LRTB* and *ROUTE-LR* inside R'. Then, we present our algorithm for the layer assignment.

Lemma 11. *The 2-layout constructed by procedures ROUTE-LRTB and ROUTE-LR inside R' satisfies the following properties.*

1. *Each wire consists of at most three segments. There are at most two hw-segments, and at most two vw-segments in each wire.*
2. *Each track (column) has hw-segments (vw-segments) from at most two nets.*

Proof. The proof is straightforward and is therefore omitted. □

Our procedure begins by assigning to three layers all the *hw*-segments, and then to four layers all the *vw*-segments. The *hw*-segments are assigned to layers by constructing a multigraph, and then coloring the edges of the multigraph. The *vw*-segments are assigned by considering each column at a time.

We construct a multigraph $G_{W'}$ from the 2-layout constructed by procedures *ROUTE-LRTB* and *ROUTE-LR* inside R'. Each track in the 2-layout is represented by a vertex in the multigraph $G_{W'}$. For each net with *hw*-segments assigned to the tracks represented by vertices v and w there is a distinct edge between vertices v and w. Note that there may be multiple edges between a pair of vertices in the graph. Since each net has at most two *hw*-segments in different tracks, and there are at most two *hw*-segments from two different nets in any track (Lemma 11), there can be at most two edges incident at any vertex in $G_{W'}$, and for each net there is at most one edge in $G_{W'}$.

A *coloration* of the edges of the multigraph is a function that assigns one of three colors (a, b, c) to each edge of the multigraph so that all edges incident on a vertex are colored differently. Since each node is of degree at most two, $G_{W'}$ can be colored in linear time (with respect to the number of nodes and edges) with the three colors. Each color corresponds to a horizontal layer, so the *hw*-segments of the net represented by an edge colored i is assigned to the layer i. The *hw*-segments of the remaining nets are assigned to horizontal layers in such a way that no two wire segments that overlap in a track are assigned to the same layer. By Lemma 11 we know that this is always possible.

The layer assignment for the *vw*-segments is performed by scanning the columns from right to left. In each column, the general rule for vertical layer assignment is that each *vw*-segment is assigned to a layer adjacent to the horizontal layer assigned to its adjacent *hw*-segment(s). Since all the *hw*-segments in a wire are assigned to one layer, the *vw*-segment can be assigned to one of two different layers. By Lemma 11, we know that each column has at most two *vw*-segments, and since there are two vertical layers adjacent to any horizontal layer, we know that this assignment is always possible. We formalize our results in Lemma 12.

Lemma 12. *Our procedure generates a seven-layer assignment for all the wire segments introduced by procedures ROUTE-LRTB and ROUTE-LR inside R' in $O(n)$ time.*

Proof. The proof follows from the above discussion and Lemma 11. □

4.2. Layer Assignment for Wires Introduced by ROUTE-REM Inside R'

Before discussing our layer assignment strategy for the wire segments introduced by *ROUTE-REM* in R', we define some useful terms, and prove properties of such 2-layout. In the 2-layout for problem instance I', constructed by the algorithm in Sec. 3, each neighbor net is either wired by an L shaped wire or by a monotone HVHV wire (i.e., a wire consisting of a horizontal segment attached to the left or right boundary, followed by a vertical segment, a horizontal segment and a vertical segment; and with the property that any vertical or horizontal non-grid line intersects the wire at most once). In the former case the net is called an *L-net*, and in the latter case it is called a *switching net (s-net)*. Note that our definitions of an L-net and an s-net are particular to a given 2-layout, i.e., the net which is an s-net in a 2-layout may not be so in another 2-layout. An s-net with a terminal point located on the left (right) boundary is called an *Ls-net (Rs-net)*. The track (column) where an s-net switches from one column (track) to the next is called an *s-track (s-column)*. The other track (column) where an *Ls*-net is routed is called the *beginning track or b-track* (*ending column or e-column*), and for an *Rs*-net it is called the *ending track or e-track* (*beginning column or b-column*). An *Rs*-net is said to be an *RBs-net (RTs-net)*, if it has a terminal located on the bottom (top)

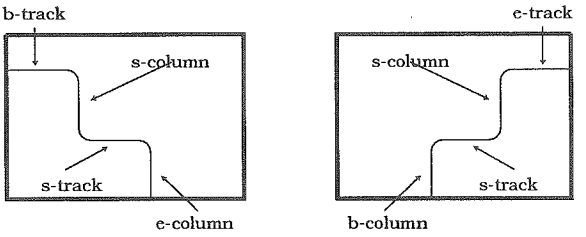


Fig. 23. *s*-net.

boundary. We define *LBs-nets and LTs-nets* similarly. Figure 23 illustrates the above definitions for an *s*-net.

An *L*-net with a terminal point located on the left (right) boundary of R' is called an *LL-net (RL-net)*. The track where an *LL*-net (*RL*-net) is routed is called the *beginning track or b-track* (*ending track or e-track*) and the column where an *LL*-net (*RL*-net) is routed is called the *ending column or e-column* (*beginning column or b-column*). Remember that these definitions are particular to a given 2-layout. The leftmost (rightmost) column with a terminal point from a net in the set N'_{TB} is called the *beginning column or b-column* (*ending column or e-column*) of the net. Let us now establish some important properties for the 2-layouts constructed by *ROUTE-REM*.

Lemma 13. *The 2-layout constructed by procedure ROUTE-REM satisfies the following nine properties.*

1. *The s-column of an Ls-net (Rs-net) is the b-column (e-column) of a non-neighbor net uniquely associated with the s-net. The vertical wires of these two nets may overlap in at most one point in this column (i.e., they may form a knock-knee at that point).*
2. *Each column has vw-segments from at most two s-nets. In particular, if it is the s-column of two s-nets, then both are Rs-nets, or both are Ls-nets.*
3. *If a column is the s-column of two Rs-nets, then one is an RTs-net, and the other is an RBs-net. Furthermore, the e-track of the RBs-net is above the e-track of the RTs-net.*
4. *The s-track of an Ls-net (Rs-net) is the e-track (b-track) of an RL-net (LL-net) uniquely associated with the s-net. The horizontal wires of these two nets may overlap in at most one point in this track (i.e., they may form a knock-knee at that point).*
5. *Each track has hw-segments from at most two s-nets.*
6. *Ech grid point has at most one knock-knee.*
7. *Procedure ROUTE-REM introduces vw-segments in each column at most twice, and each time either one or two wire segments are added. When*

two segments are added in a single operation, then at most one of them belongs to an s-net.

8. *If a column has a trivial net, then the column does not have any other vw-segments.*
9. *The nets associated with two s-nets with the same s-column, do not overlap with each other in that column.*

Proof. Proof of (1): When procedure *ROUTE-REM* introduces an s-net, the switching takes place in order to accommodate the start of a net in the 0_n *run* at the s-column (see Fig. 16(c)), or to set up a new empty horizontal track that is necessary for the next step of the routing algorithm (see Fig. 18). Thus there is a net uniquely associated with the s-net that starts (ends) in the s-column of an Ls-net (Rs-net). Furthermore, by construction, we know that the two nets may overlap in at most one point in the s-column.

Proof of (2): Since the s-column of every s-net is the b-column or e-column of a non-neighbor net uniquely associated with the s-net (1), we associate the terminal point of the non-neighbor net located in the s-column with the s-net. By definition of an s-net, the b-column or e-column of an s-net has a terminal point from the s-net. Therefore, each of the two columns where an s-net is routed has a terminal point uniquely associated with the s-net. Since there are at most two terminal points in each column, it must be that there are vw-segments from at most two s-nets in a column.

When an Rs-net (Ls-net) is introduced by procedure *ROUTE-REM*, it is introduced on the right (left) side of the current rectangle G'. In order for an Rs-net and an Ls-net to have the same s-column, the column should be the right side of a rectangle G', and the left side of another rectangle G'. But, since each new rectangle G' is enclosed by the previous rectangle, and G' is never a single line segment, an Rs-net and an Ls-net cannot have the same s-column.

Proof of (3): From (1), we know that for each s-net there is a distinct net with a terminal point in the s-column of the s-net. Since there is at most one terminal point on the top and bottom ends of a column, the two Rs-nets with the same s-column must be an RTs-net and an RBs-net.

An RTs-net (RBs-net) is introduced by procedure *ROUTE-REM*, when the doubly covered corner is located on the bottom (top) right corner of the rectangle G'. Since each time an s-net is introduced, the new rectangle G' is enclosed within the previous rectangle G', it must be that the e-track of the RBs-net is above the e-track of the RTs-net.

Proof of (4): When procedure *ROUTE-REM* introduces an Ls-net (Rs-net), the s-track of the Ls-net (Rs-net) is the e-track (b-track) of an RL-net (LL-net) [see Figs. 16(c) and 18] uniquely associated with the Ls-net (Rs-net). Furthermore, the hw-segments of the two nets may overlap in at most one point in this track.

Proof of (5): Since the s-track of every s-net is the b-track or e-track of an L-net uniquely associated with the s-net (4), we associate the terminal point of

the L-net located in the s-track with the s-net. By the definition of an s-net, the b-track or e-track of an s-net has a terminal point from the s-net. Therefore, each of the two tracks where an s-net is routed has a terminal point uniquely associated with the s-net. Since there are at most two terminal points in each track, there are hw-segments from at most two s-nets in any track.

Proof of (6): Whenever procedure *ROUTE-REM* introduces a knock-knee at point p, at least one of the terminal points in that column belongs to the nets that form the knock-knee. Since each column has at most two terminal points, it must be that if p has two knock-knees, then the two terminals in that column belong to distinct nets, one from each knock-knee. Suppose that procedure *ROUTE-REM* introduces two knock-knees at grid point p. Let G'' be the rectangle G' in procedure *ROUTE-REM* when the first knock-knee was introduced at grid point p. If p is not a corner of G'', then we know that when the knock-knee was introduced at p, both the terminal points in the column of p belong to the nets in the knock-knee. Since every time a knock-knee is introduced by procedure *ROUTE-REM* at point p, at least one terminal point in column p belongs to the nets in the knock-knee, it cannot be that another knock-knee can be introduced at p. A contradiction, hence it must be that p was introduced at a corner c of G''. Assume that c is the top-left corner of G''. In this case, the terminal points "closest" to corner c (i.e., the one exactly above and exactly to the left of c) belong to the nets in the knock-knee. By arguments similar to the ones above, we know that the next knock-knee at p is introduced at a corner c' of a new rectangle G' (which we call new G''). Since the terminal point "closest" to corner c' belongs to one of the nets whose wires form the knock-knee, it must be that c' is the bottom-right corner of G''. Since at each iteration the new G' is within the previous G', we know that the new G'' must be a single point. But then, no knock-knee is introduced. A contradiction. Hence each grid point contains at most one knock-knee. This completes the proof of (6).

Proof of (7)–(9): The proof of (7) is straightforward. Every time procedure *ROUTE-REM* introduces a vw-segment, it also routes a net with a terminal point in that column. Therefore, if a column has a trivial net, then the column does not have any other vw-segments, and statement (8) holds. Statement (9) follows directly from (3). This completes the proof of (7)–(9) and the lemma. □

Our layer assignment procedure for the wire segments introduced by procedure *ROUTE-REM* for I' begins by assigning to three layers all the hw-segments and then to four layers all the vw-segments. One can easily assign all the hw-segments to three layers by considering each track at a time. However, this arbitrary assignment may not always lead to a feasible assignment of the vw-segments. In order to be able to generate quickly the vertical layer assignment, our horizontal layer assignment must satisfy some special properties. To perform the layer assignment of the hw-segments, we partition the set of s-nets in N' into four classes, such that the fourth class is a subset of the Rs-nets. Then the hw-segments of all the nets in N' are assigned to layers a, b, and c, in such a way that all hw-segments of each net are

assigned to the same layer, except for nets in the fourth class. The *hw*-segments of each fourth class *Rs*-net are assigned to layers *a* and *b*, or *c* and *b*. The *vw*-segments are assigned to layers 1, 2, 3, or 4 by scanning the columns from left to right. The general vertical layer assignment rule is as follows: each *vw*-segment is assigned to a layer adjacent to the horizontal layer(s) where its adjacent *hw*-segment(s) have been assigned.

4.2.1. Horizontal layer assignment

Before we present our horizontal layer assignment procedure *HLA*, we discuss our subprocedure that partitions the *s*-nets in N' into four classes, such that the fourth class contains only *Rs*-nets. This procedure begins by partitioning the *hw*-segments in each track into three groups; next, it constructs a multigraph, and then colors the edges in the multigraph with four colors. Let us now discuss these two steps in detail.

Since each grid edge contains at most two wire segments, and there is at most one knock-knee at each grid point (Lemma 13(6)), we know that all *hw*-segments in each track can be partitioned into three groups, such that the wire segments assigned to the same group do not overlap (not even at a single point). Furthermore, the partitioning of the segments is generated in linear time by scanning the track from left to right. Later on we may slightly modify this partition. Since an *s*-net has two *hw*-segments in different tracks, its *hw*-segments belong to groups in two different tracks. Obviously, the *hw*-segments of two *s*-nets assigned to a common track, may either belong to the same group or not.

Next, we construct a multigraph $G_{W'}$ from the 2-layout W' of I' generated by procedure *ROUTE-REM*. Each track in the 2-layout is represented by a vertex in the multigraph $G_{W'}$. For each *s*-net with *hw*-segments assigned to the tracks represented by vertices v and w there is a distinct edge between vertices v and w. Note that there may be multiple edges between a pair of vertices in the graph. Since each *s*-net has two *hw*-segments in different tracks, and there are at most two *hw*-segments from two different *s*-nets in any track (Lemma 13(5)), there can be at most two edges incident at any vertex in $G_{W'}$, and for each *s*-net there is exactly one edge in $G_{W'}$. The vertex corresponding to a track is labeled *d* if the *hw*-segments from two *s*-nets assigned to this track are in different groups (in the partitioning of the *hw*-segments in this track); otherwise, it is labeled *s*. Note that every vertex is labeled.

By Lemma 13(5) we know that the degree of each vertex is at most two. Therefore, the connected components of a multigraph $G_{W'}$ are either chains each with at least one vertex (Fig. 24(a)), or cycles (Fig. 24(b)). The tracks (*s*-nets) associated with a connected component of a multigraph are the tracks (*s*-nets) represented by the vertices (edges) in the component. The nets associated with a connected component are all of those nets with a *hw*-segment located on a track associated with the component.

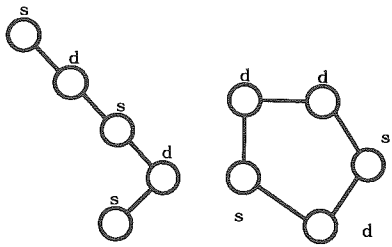


Fig. 24. Multigraph $G_{W'}$.

In order to wire our layout in seven layers we need to treat separately the components formed by a cycle with exactly one vertex labeled d. These cycles are called *single-d cycles*. What we would like to establish is that just before the first Rs-net in the single-d cycle was introduced by *ROUTE-REM*, the topmost and bottommost tracks represented by the vertices in the single-d cycle had at most one hw-segment from the s-nets in the cycle. We call this property the *ordering property*. In general one cannot prove that every single-d cycle satisfies the ordering property; however, in what follows we shall modify slightly the previously generated partitions of the horizontal tracks so that all the resulting single-d cycles in the multigraph satisfy the ordering property and the track partition is valid.

For each single-d cycle that does not satisfy the ordering property we perform the following operations. By definition the number of s-nets in the cycle must be at least two. If the first or second s-net routed by *ROUTE-REM* is an Rs-net, then it is simple to see that it satisfies the ordering property. So let us consider the remaining case, i.e., the first two s-nets in the single-d cycle routed by *ROUTE-REM* are Ls-nets. Let x and y be these two s-nets, and assume net x was routed by *ROUTE-REM* before net y. One of these nets is an LBs-net and the other is an LTs-net, as otherwise the component does not form a cycle. Suppose that the s-track of one of these two nets is the b-track of the other. If the s-columns of nets x and y are the same, then either the two b-tracks are labeled d, in which case the cycle is not a single-d cycle; or the horizontal wire segment in the b-track of one of these nets overlaps with at most one other hw-segment and the track is labeled s, in which case the wire can be assigned to a different group in the partition of that track and the new cycle has two vertices labeled d. If the s-columns of x and y are different, then the s-column of x is to the left of the s-column of y. It is simple to verify that the hw-segment of net x located in its b-track overlaps with at most another horizontal segment and can be reassigned in the partitioning of the segments in that track so that the new cycle has either zero or two vertices labeled d. In some cases the segment just to the right of the segment reassigned may also need to be reassigned. So we need to consider only the case when the s-tracks of these s-nets (x and y) is not the same as the b-track of the other, and one of these

s-nets is an LBs-net and the other is an LTs-net. Since the two b-tracks have only one wire from the s-nets in the cycle, there must be at least three s-nets in the cycle. If the next s-net from the simple-d cycle routed by $ROUTE - REM$ is an Ls-net, then its b-track must be different than the b-tracks of x and y. Since the remaining s-nets in the cycle will only be introduced inside the G' at this point, it then follows that the component does not form a cycle. So this third net must be an Rs-net, and by the above discussion the simple-d cycle satisfies the ordering property. Therefore, after performing the above transformations all the single-d cycles in the resulting multigraph satisfy the ordering property.

A *coloration* of the edges of the multigraph is a function that assigns one of four colors to each edge of the multigraph so that each edge is labeled with one of the first three colors, and for each degree-two vertex labeled s (d) the two edges incident to it are colored identically (differently). The only exceptions to this rule are the single-d cycles, in which case one of the edges representing an Rs-net is labeled with the fourth color and the vertices labeled s adjacent to this edge do not need to satisfy the property that their edges are colored with the same color.

Let us now outline our procedure to color the edges of the multigraph with four colors. First, each chain is transformed into a cycle by adding two (dummy) vertices labeled d adjacent to each other, and each one adjacent to one of the end vertices in the chain. After this operation all the components are cycles and each cycle has at least one vertex labeled d. Each non single-d cycle is transformed as follows. Each path from v to w with all vertices labeled s except for v and w is replaced by a single edge adjacent to v and w (all nodes labeled s in the path are deleted). As a result of this, all these cycles have at least two vertices and all their vertices are labeled d. It is simple to show that these cycles can be easily colored with the first three colors. The edges in the original components corresponding to these cycles are colored with the same color as the edge that replaced them. The dummy vertices and edges are then deleted. The cycles with exactly one vertex labeled d are colored differently. First we find the Rs-net with the rightmost s-column amongst all the Rs-nets represented by the edges in the cycle. The edge corresponding to this Rs-net is assigned the fourth color. Relabel the vertices adjacent to this edge d and color the other edges in the cycle by the procedure used for the case when the cycle is not a single-d cycle.

We classify the s-nets by assigning all s-nets corresponding to edges colored i to the i^{th} class. Procedure HLA for the layer assignment of the hw-segments based on this classification of the s-nets is defined below.

procedure HLA
Partition the hw-segments in each track into three groups;
Construct the multigraph $G_{W'}$;
Modify slightly the partition as described earlier so that all the remaining single-d cycles in the resulting multigraph $G_{W'}$ satisfy the ordering property;
Color the edges of the multigraph edges using the procedure discussed above.

Assign the s-nets represented by edges colored i to the i^{th} class.
while there are unassigned hw-segments of fourth class s-nets **do**
$k \leftarrow$ rightmost s-column of a fourth class s-net with an unassigned hw-segment.
/* In Lemma 14(1) we prove that all the hw-segments of fourth class nets whose s-column is column k have not been assigned. */
Let n'_i be a fourth class s-net with its s-column in column k;
Let C be the component of n'_i.
Let $n'_{i'}$ be the net associated with n'_i. /* the net identified in Lemma 13(1) */
/* Net n'_i is represented by an edge in $G_{W'}$ that is part of a single-d cycle. */
/* In what follows we assign to layers the hw-segments of a subset of nets that includes n'_i and possibly $n'_{i'}$, depending on the vw-segments in column k. */
There are two cases depending on the vw-segments in column k.
Case 1: The only vw-segments in column k belong to nets n_i' and $n'_{i'}$.
Assign the hw-segments of n'_i in the s-track (e-track) to layer a (b).
Map the first three colors to the layers consistent with the assignment of n'_i.
Assign to layers the hw-segments of the remaining s-nets associated with component C. The assignment must be consistent with the above mapping.
For each track t associated with component C, assign to layers the remaining hw-segments. The assignment must be consistent with the assignment defined so far and the previously constructed grouping (first step) for track t.
/* Note that the hw-segment of net $n'_{i'}$ might not be assigned. */
Case 2: There are three or four vw-segments in column k.
Let n'_j be the other s-net with vw-segments in column k, unless there are no other s-nets with this property in which case n'_j is the net other than n'_i and $n'_{i'}$ with vw-segment in column k.
If the hw-segment(s) of n'_j has not been assigned, and if n'_j is not an s-net whose vw-segment overlaps (in more than one point) with that of $n'_{i'}$ in column k, **then** Assign the hw-segment of net n'_j to layer b, unless n'_j is a fourth class s-net in which case assign the hw-segments of n'_j in the e-track and s-track to layers b and c, respectively.
Let C' be the component where n'_j belongs.
Assign all the hw-segments in all tracks in C' as in Case 1.
If the hw-segments of n'_j have not been assigned, or have been assigned to either assigned to either layer b or c, **then** execute the steps of Case 1.
Otherwise, execute the steps of Case 1, but the hw-segments of n'_i are assigned to layers c and b rather than to layers a and b.
end while
for each component C with an unassigned hw-segments in one of its tracks **do**
/* In Lemma 14(2) we show that all the hw-segments of the nets associated with component C are not assigned. */
Let M be any mapping of the first three colors to the horizontal layers.
Proceed as in Case 1 to assign to layers all the hw-segments of the nets in C.
end for
end of procedure HLA

Figure 25 illustrates the layer assignment of the *hw*-segments of the wires in the routing layout for I' given in Fig. 21. Lemma 14 establishes the correctness of procedure *HLA*, and Lemma 15 establishes some important properties which we use to show that the vertical layer assignment is feasible.

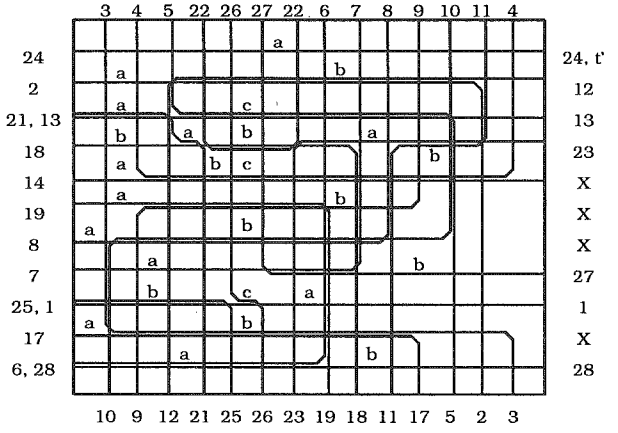


Fig. 25. Layer assignment of all *hw*-segments.

Lemma 14.

(a) At the beginning of each iteration of the while-loop and the for-loop of procedure HLA, and at the end of procedure HLA, the following statements hold.

1. *There is a vertical line l that partitions R' into two rectangles such that all the hw-segments of the fourth class s-nets in the right (left) rectangle have been assigned (not been assigned) to layers. Furthermore, at the beginning of the i^{th} iteration of the while-loop, the right rectangle of R' defined by l includes only the rightmost $i-1$ s-columns of fourth class nets.*
2. *Either all or none of the hw-segments of nets associated with a component have been assigned to layers.*

(b) At the end of the while-loop of procedure HLA, (1) holds with the right rectangle being the entire rectangle R'.

(c) At the end of the procedure HLA, all the hw-segments of W' have been assigned to layers.

Proof. Clearly (a) holds at the beginning of the while-loop. Let us now show that if (a) holds at the beginning of the while-loop, then it holds at the end of the loop. At each iteration we assign all the *hw*-segments in the tracks associated with

the components that include the fourth class nets with their s-column in column k and possibly those in another component. All the components whose tracks are assigned in this iteration include at least one net with a vw-segment in column k. By construction each component has at most one fourth class net. Therefore, to complete the proof we need to show that the components assigned in this iteration either include a fourth class net with its s-column in column k, or do not include a fourth class net. Suppose this is false. Suppose that component C assigned in this iteration has a fourth class net with its s-column to the left of column k. This component must include net n'_j which is not an s-net. Net n'_j was introduced by $ROURE - REM$ when some s-net x in C was routed. Since the vw-segments of nets n'_j and n'_i, overlap in column k and both of this nets have a terminal point at that column, it must be that the track with a terminal point from net x is above (below) the e track of n'_i when x has a terminal point on the bottom (top) side of R'. It cannot be that x is an Rs-net, as otherwise its s-column would be to the right of column k. So x is an Ls-net. Since the cycle satisfies the ordering property it must be that when the first Rs-net in the cycle is routed by $ROUTE\text{-}REM$, there is only one wire segment in the b-track of net x from the s-nets in C. Since C forms a cycle, such a track must contain two hw-segments from the s-nets in the cycle. So this Rs-net is introduced before n'_i, or the component is not a cycle. A contradiction. Therefore (a) holds at the end of the first loop of procedure HLA. Since (a) holds at each iteration, (b) holds. The proof for (c) is straight forward and thus omitted. □

4.2.2. Vertical layer assignment

First, in Lemma 15, we establish some properties of the horizontal layer assignment generated by procedure HLA, which are used in the vertical layer assignment procedure.

Lemma 15. *The following statements hold after procedure HLA.*

1. *Each column has wires from at most two s-nets that belong to the fourth class.*
2. *The hw-segments of each fourth class net are assigned to layers a and b, or c and b.*
3. *Any three hw-segments of three distinct nets adjacent to three vw-segments that overlap at a grid point, have not been assigned to the same layer.*

Proof. By Lemma 13(2) and the fact that each net represented by a vertex colored with the fourth color is an Rs-net, we know that there can be no more than two s-nets in any column. Therefore, statement (1) holds. From procedure HLA, we know that the hw-segments of each net from the fourth class are assigned to layers a and b (or c and b). Therefore, statement (2) holds. When three vw-segments overlap at a grid point, two of these vw-segments belong to two nets that

form a knock-knee at that grid point. By procedure HLA, the two *hw*-segments that form the knock-knee are assigned to different layers. If both nets that form the knock-knee do not belong to the fourth class, then clearly statement (3) holds. Otherwise, at most one of them is a fourth class *s*-net. From Case 2 of procedure HLA, we know that three *hw*-segments of these three nets have not been assigned to the same layer. Therefore, statement (3) holds. □

Next, we explain the layer assignment of the *vw*-segments (procedure VLA). As mentioned before, the *vw*-segments are assigned to one of four layers, and the layer assignment is performed by scanning the columns from left to right. In each column, the general rule for vertical layer assignment is: each *vw*-segment is assigned to a layer adjacent to each horizontal layer assigned to its adjacent *hw*-segment(s). For example, if the *hw*-segments of a net are assigned to layers a and b, then the *vw*-segment adjacent to both of these segments is assigned to layer 2. On the other hand, if the *hw*-segment(s) is (are) assigned to layer a, then the *vw*-segment can be assigned to layer 1 or layer 2. In what follows, we show that there is a feasible layer assignment consistent with this general rule, and that such assignment can be generated quickly. A feasible assignment is one that satisfies the multilayer wiring conditions given in Sec. 2. There are three cases depending on the number of *vw*-segments in the column.

Case 1: There are at most two *vw*-segments in the column.

Each *vw*-segment is assigned to a layer by following the general rule together with the restriction that both segments are assigned to different layers. Since there are at most two *vw*-segments of two nets, at most one belongs to an *s*-net (Lemma 13(7)). Therefore, a feasible assignment exists.

Case 2: There are three *vw*-segments in the column.

Each *vw*-segment is assigned to a layer by following the general rule together with the restriction that no two segments that overlap are assigned to the same layer. By Lemma 15(3), we know that no three *hw*-segments of distinct nets that are adjacent to *vw*-segments that overlap at a grid point are assigned to the same layer. If one of the *vw*-segments belongs to a fourth class net, then it is assigned to layer 2 (3) if the *hw*-segments of the fourth class net are assigned to layers a and b (c and b). One can easily show that the remaining *vw*-segments can be assigned to layers as per the general rule. On the other hand, it is simple to show that a feasible assignment exists if not all the *hw*-segments adjacent to the *vw*-segments in this column are assigned to the same layer. Otherwise, we know that no two of them are part of a knock-knee in this column. Assume without loss of generality that all the *hw*-segments adjacent to the *vw*-segments are assigned to a. Since there is at most a two-overlap, it must be that at least two of the three *vw*-segments do not overlap. These wire segments are assigned to layer 1, and the third is assigned to layer 2. Therefore, a feasible assignment consistent with our rules exists. Figure 26 shows the layer assignment of the leftmost three columns of the 2-layout of Fig. 21,

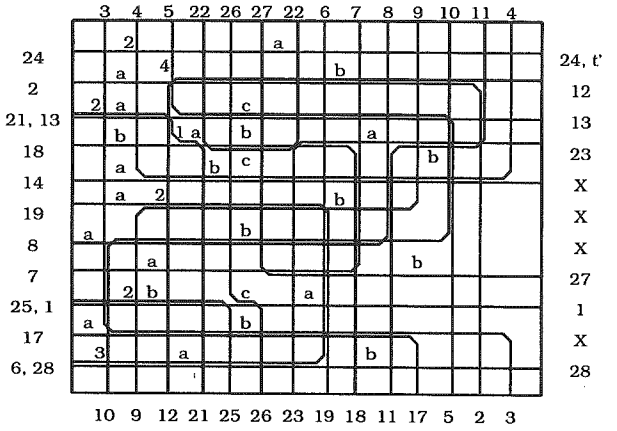


Fig. 26. Layer assignment for wire segments of three columns.

the first two of which are assigned as described in Case 1, and the third column is assigned as described in Case 2.

Case 3: There are four *vw*-segments in the column.

Each *vw*-segment is assigned to a layer by following the general rule together with the restriction that no two segments that overlap are assigned to the same layer. By Lemma 15(3), we know that no three *hw*-segments of distinct nets that are adjacent to *vw*-segments that overlap at a grid point are assigned to the same layer. This implies that all *vw*-segments in this column that are adjacent to *hw*-segments assigned to the same layer can be partitioned into two groups, each of which can be assigned to one of the vertical layers adjacent to the horizontal layer. Hence we can assign the *vw*-segments to layers adjacent to the horizontal layers where the *hw*-segments adjacent to these *vw*-segments are assigned, i.e., the vertical layer assignment is consistent with the general vertical layer assignment rule.

By Lemma 13(7), these three cases include all possibilities of the number of *vw*-segments in a non-empty column. The above process is repeated until all non-empty columns have been considered, at which point procedure *VLA* terminates.

From the above procedures we know that all the *vw*-segments (*hw*-segments) were assigned to the layers 1, 2, 3, and 4 (*a*, *b*, and *c*). Figure 27 illustrates the layer assignment of all the nets for the layout for problem I' given in Fig. 21.

Lemma 16. *Procedures HLA and VLA generate a feasible seven-layer assignment for all the wire segments in* R' *in* $O(n)$ *time.*

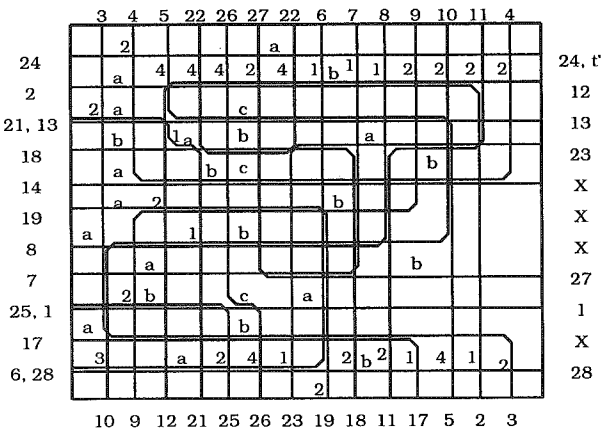


Fig. 27. Layer assignment of problem I'.

Proof. It is simple to show that all the steps in procedures HLA and VLA can be implemented to take $O(n)$ time. The remaining part of the proof follows from Lemmas 13, 14, 15, and the discussion before this lemma. □

4.3. Layer Assignment for Wires Introduced by ROUTE-LRTB Outside R'

Let us now consider the layer assignment for the wire segments introduced by *ROUTE-LRTB* outside R'. We consider only the wire segments located to the left of the leftmost vertical stretching line, since the wire segments to the right of the rightmost stretching line can be handled similarly. We shall refer to this rectangle as R''. Before we present our layer assignment strategy, we define some terms, and prove some properties of the 2-layouts generated by *ROUTE-LRTB* in R''. It is important to note that the additional column is needed in order for the *hw*-segments to change layers. If this column is not introduced, then it is impossible to perform the layering without making major changes to the layer assignment of R'. To convince yourself of this fact try to wire the column with nets 1 and 7 in Fig. 28 without allowing a change of layers in the layer assignment of R'.

Procedure *ROUTE-LRTB* introduces wires in R'' with at most three wire segments, of which at most two are *hw*-segments and at most two are vertical (Lemma 17). A net is called an *s*-net if it contains three wire segments in R'', and two of them are horizontal. Note that our definition of an *s*-net is particular to a given 2-layout, i.e., the net which is an *s*-net in a 2-layout may not be so in another 2-layout. When *ROUTE-LRTB* in R'' routes an *s*-net, it is because the *s*-net includes a terminal point on the left boundary of R'', and which is located on the track corresponding to the upper or lower boundaries of rectangle G. The tracks where an *s*-net has its *hw*-segments are called the *b*-track and the *e*-track. The *b*-track is

the one that includes the above terminal point. The s-column of an s-net is the column where the s-net has its vw-segment. The corner of the wire for an s-net in its b-track is marked upper (lower) if the b-track is the track corresponding to the upper (lower) boundary of G. A t-net associated with an s-net is a net connected by a vertical-horizontal wire or a vertical-horizontal-vertical wire, whose leftmost terminal point is in the s-column of the s-net, and whose hw-segments are not in any of the interior tracks of the rectangle G defined when the s-net was introduced.

Our layer assignment procedure begins by assigning the hw-segments intersecting the stretched column to layers a or c. This assignment should be consistent with the layer assignment generated in Subsections (4.1) and (4.2). We shall refer to this assignment as a tentative assignment. This assignment may change (i.e. a could become b or c could become b, but not both) and eventually becomes the final assignment. Then, the procedure performs the layer assignment of all the remaining wire segments while scanning the columns in R'' from right to left. At each column, we define an upper track and a lower track. At the rightmost column of the rectangle R'', the upper (lower) track is one of the interior tracks of the last rectangle G defined in procedure *ROUTE-LRTB*. In case the last G has only two tracks, then the upper and lower tracks are a hypothetical track between the two tracks of the rectangle G. At column k, the upper track is the topmost track where the marked upper corner of an s-net with s-column k is located, unless there are no such nets in which case the upper track is the same as the upper track of column $k+1$ (i.e., the previous column). The lower track is defined similarly. All the hw-segments with a point to the right of column k, and between (inclusive) the upper and lower tracks of column $k+1$ have the final assignment. The remaining hw-segments have a temporary assignment. Then, at each iteration we make a final assignment of all the hw-segments in the tracks between (inclusive) the upper and lower tracks of the current column, with a point to the right of the current column. All the hw-segments assigned to layers in the temporary assignment are assigned to layers a or c. One of these assignments may change to b. Therefore, whenever we make a temporary assignment we should make sure that it is possible to change it to b. Before we present our procedure, we establish some properties of the 2-layout generated by *ROUTE-LRTB* in R''.

Lemma 17. *The 2-layout constructed by procedure ROUTE-LRTB in R'' satisfies the following properties.*

1. *Each wire consists of at most three segments. There are at most two hw-segments and at most two vw-segments in each wire.*
2. *Each wire with two horizontal segments is an s-net and it has a terminal point on the left boundary of rectangle R''. In the s-column of an s-net there could be a vw-segment of a t-net. Either these two nets form a knock-knee or their vw-segments do not overlap.*
3. *At most two s-nets have the same s-column.*

4. *There are at most two sets of nets with vw-segments in a column. Each set contains at most two nets, and if there are two nets, then one is an s-net and the other is a t-net.*
5. *If there are two s-nets at a column then the vw-segments of one of the s-nets may overlap (at more than one point) with the vw-segments of the t-net associated with the other s-net; however, both nets do not satisfy this property.*

Proof. The proof is straightforward and is therefore omitted. □

It is simple to show that if there is an s-net with its corner point marked upper (lower) at a column, then the b-track of the s-net is above (below) the upper (lower) track in the column immediately to its right. Let us now discuss our procedure to perform the layer assignment when considering column k. There are three cases depending on the number of s-nets with vw-segments at column k.

Case 1: There are no s-nets with s-column at column k.

By Lemma 17(4) and the conditions of the case, we know that there are vw-segments from at most two nets at column k and neither of the nets is an s-net. Therefore, the upper (lower) track at column k is equal to the one at column $k+1$ (i.e. the previous column). Let x and y be the two nets with vw-segments in column k. In case there is only one net, ignore the operations for net y. If the hw-segments of net x (y) have not been assigned, then assign them to layer a (c) and their vw-segment in column k to layer 2 (3). Note that this allows us to change them to b later on, if they are still temporarily assigned. Assign the remaining vw-segments to layers consistent with the previous assignments. This is always possible since there are two vertical layers adjacent to every horizontal layer.

Case 2: There is only one s-net with s-column at column k.

By Lemma 17(4) we know that there are vw-segments from at most three nets. By the conditions of the case, we know that there is an s-net and its corresponding t-net, and another net (x). Assign the hw-segment of the s-net in the b-track to the same layer as the hw-segment in its e-track, unless it has not been assigned, in which case both are assigned to layer a or layer c. As a result of this assignment, a temporary assignment of hw-segments located in the b-track of the s-net may have to be changed to layer b. Note that after this assignment, that track will have its final assignment. If the hw-segment of net x has not been assigned, then assign it to layer a. Assign its vw-segment to either layer 2 or 3, consistent with the assignment of its hw-segment. Now assign the vw-segments of the s and t nets. This assignment is always possible, since all the hw-segments in each of these two nets have been assigned to the same layer and there are two vertical layers adjacent to every horizontal layer.

Case 3: There are two s-nets with their s-column at column k.

By Lemma 17(4), we know that there are vw-segments from at most four nets, two pairs of s-t nets. Perform the horizontal layer assignment for each pair, as

in Case 2. By Lemma 17(5), we know that there is at most one s-net whose vw-segment overlaps at more than one point with that of the t-net in the other pair. Let s_1 be that s-net, unless no such net exists, in which case s_1 is either of the two s-nets. Assign the vw-segment of the net s_1 to layer 2 or 3, depending on the layer where its hw-segment have been assigned. The assignment of the other vw-segments proceeds as in Case 2.

As mentioned before, the above process is repeated on the other side of rectangle R'. Figure 28 illustrates how our procedure handles the layer assignment of the first three columns of R. Cases 3, 2, and 1 arise when considering the third, second, and first columns of R, respectively. Lemma 18 establishes the main result in this Section.

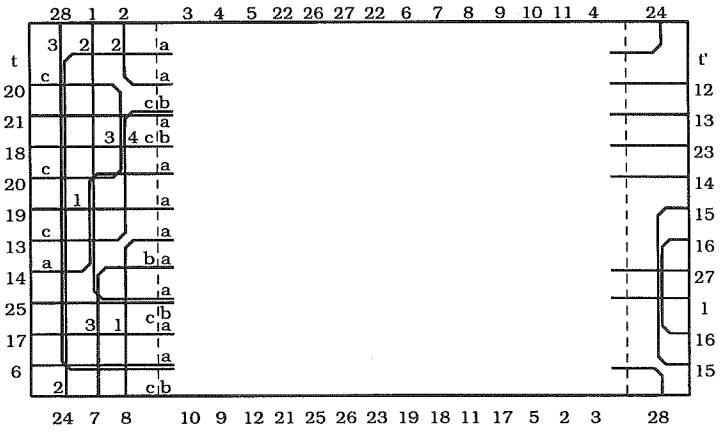


Fig. 28. Layer Assignment of wires introduced by *ROUTE-LRTB*.

Lemma 18. *Our procedure generates a feasible seven-layer assignment for all the wire segments introduced by ROUTE-LRTB inside R'' in $O(n)$ time.*

Proof. The proof follows from the above discussion and Lemma 17. □

4.4. Complete Layer Assignment

For any particular problem instance, we perform the layer assignment of Subsections (4.1) and (4.3), or (4.2) and (4.3). Figure 29 shows the complete layer assignment of all the nets for the routing layout of problem I given in Fig. 22.

Theorem 2 establishes our result in this Section, and Theorem 3 establishes the main result of this paper.

Theorem 2. *Our procedure constructs a seven-layer wiring for any 2-layout generated by procedure ROUTE. The time complexity of our algorithm is linear*

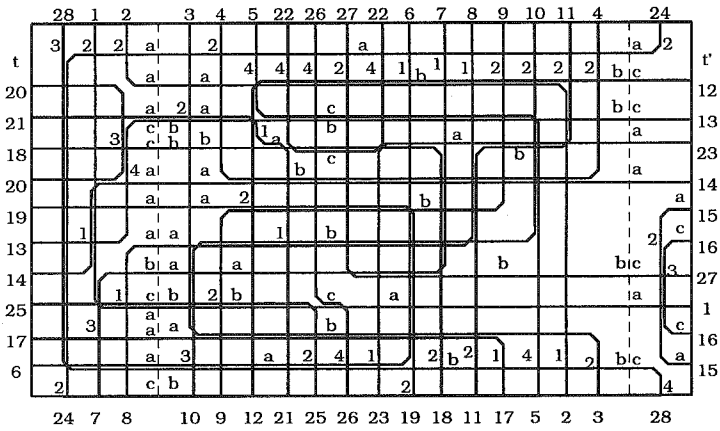


Fig. 29. Layer assignment.

with respect to n, the number of terminal points (if the set of terminal points is initially ordered). The 2-layout contains no more than $1.5v^*$ *bends where* v^* *is the natural bend number for I.*

Proof. Procedure *ROUTE-REM* is the only procedure that connects nets with wires with a number of bends that exceeds their natural bend number. By Lemma 13(1) and (4), and procedure *ROUTE-REM*, we know that for every net connected by a wire whose number of bends exceeds the natural bend number of the net, the procedure connects at least two nets by wires with a number of bends equal to the natural bend number of the nets. One of these wires has one bend and the other has two bends. Therefore, $1.5v^*$ is an upper bound on the number of bends introduced by the algorithm. The proof of the time complexity bound is straightforward. The remaining part of the proof follows from Lemmas 12, 16, and 18. □

Theorem 3. *A seven-layer wiring for any two-terminal-net two-overlap routable SBR problem can be constructed in* $O(n)$ *time, if the set of n terminals is initially ordered. The layout has at most two extra tracks and two additional columns.*

Proof. By Theorems 1 and 2. □

5. Discussions

We presented an algorithm that given any two-overlap routable *SBR* problem instance, adds at most two tracks and two columns, and wires it in seven layers. The time complexity for our algorithm is $O(n)$ when the set of n terminal points is

initially ordered. The constant associated with the time complexity bound is small. All the wires in each layer are either vertical or horizontal, but not both. Most of the vias introduced by our algorithm join wires in adjacent layers. The rest of the vias join wires in adjacent horizontal layers, and are located along two columns. It may be possible to perform the wiring in seven layers by adding only one additional track or column slot; however, such a procedure is extremely complex.

Our algorithm can be generalized to solve the multiterminal-net *SBR* problem under the two-overlap, by splitting the multiterminal nets into two-terminal nets and stretching the layout,[20,21] multiterminal nets can be partitioned into two-terminal nets. By stretching the grid of R, our algorithm can be directly applied to obtain seven-layer wiring solutions. Our algorithm can be easily adapted to solve the two-stacked pin two-overlap *SBR* problem when the set of terminals is compatible. In this case, the set of terminals is said to be compatible if for each track or column only one of the endpoints has terminal points. Any multiterminal *SBR* problem with a set of compatible terminals can be transformed to a two-stacked pin *SBR* problem by duplicating midpoints of a multiterminal net. Our algorithm can be modified to handle this problem. We have also generalized our algorithm to handle higher number of overlappings. For brevity, we do not discuss further generalizations of our algorithm.

References

1. M. Sarrafzadeh, "Channel-routing problem in the knock-knee mode is NP-Complete", *IEEE Transactions on Computer Aided Design* **CAD-6(4)** (1987) 293–303.
2. T. G. Szymanski, "Dogleg channel routing is NP-complete", *IEEE Transactions on Computer-Aided Design* **4(1)** (1985) 31–41.
3. B. S. Baker, S. N. Bhatt and F. T. Leighton, "An approximation algorithm for Manhattan routing", *Advances in Computer Research*, Ed. F. P. Preparata, **2** (1984) 205–229.
4. M. Burstein and P. Pelavin, "Hierarchical wire routing", *IEEE Transactions on Computer-Aided Design* **CAD-2(4)** (1983) 223–234.
5. D. N. Deutsch, "A Dogleg channel router", Proceedings of the 13th Design Automation Conference (1976) 425–433.
6. R. L. Rivest, A. E. Baratz and G. Miller, " Provably good channel routing algorithms", in *VLSI Systems and Computations*, ed. H. T. Kung, R. Sproull and G. Steele, Computer Science Press, Rockville, Maryland (1981).
7. J. Soukup and J. C. Royle, "On hierarchical routing", *J. Digital Systems* **5(3)** (1981).
8. T. Yoshimura and E. Kuh, "Efficient algorithms for channel routing", *IEEE Transactions on Computer-Aided Design* **CAD-1(1)** (1982) 25–35.
9. R. L. Rivest and C. M. Fiduccia, "A greedy channel router", *Proceedings of the 19th Design Automation Conference* (1982).
10. W. K. Luk, "A greedy switch-box router", *INTEGRATION, the VLSI journal* **3** (1985).
11. A. Hashimoto and J. Stevens, "Wiring routing by optimizing channel assignment within large apertures", Proceedings of the 8th Design Automaton Conference (1971) 155–160.

12. T. F. Gonzalez and S. Q. Zheng, "A near-optimal algorithm for routing compatible nets inside a rectangle", UCSB Technical Report #TRCS 87-14 (1987).
13. A. Frank, "Disjoint paths in rectilinear grid", *Combinatorica* **2(4)** (1982) 361–371.
14. F. P. Preparata and W. Lipski, Jr, "Optimal three-layer channel routing", *IEEE Transaction on Computers* **33(5)** (1984).
15. K. Mehlhorn, F. P. Preparata and M. Sarrafzadeh, "Channel routing in knock-knee mode: Simplified algorithms and proofs", *Algorithmica* **2(1)** (1986) 213–221.
16. T. F. Gonzalez and S. Q. Zheng, "Simple three-layer channel routing algorithms", *Proceedings of the 3rd AEGEAN Workshop on Computing* (*AWOC 88*), Lecture Notes in VLSI Algorithms and Architectures, Springer-Verlang (1988) 237–246.
17. R. Kuchem, D. Wagner and F. Wagner, "Area optimal three layer channel routing", *Proceedings of the Symposium on Foundations of Computer Science* (1989) 506–511.
18. S. Gao and M. Kaufmann, "Channel routing of multiterminal nets", *Proceedings of the 28th Symposium on Foundations of Computer Science* (1987) 316–325.
19. C. Wieners-Lummer, "Three-layer channel routing in knock-knee mode", Technical Report, University of Paderborn, Germany.
20. K. Mehlhorn and F. Preparata, "Routing through a rectangle", *Journal of ACM* **33(1)** (1986) 60–85.
21. T. F. Gonzalez and S. Q. Zheng, "Grid stretching algorithms for routing multiterminal nets through a rectangle", *INTEGRATION: the VLSI Journal* **13(2)** (1992).
22. M. L. Brady and D. J. Brown, "VLSI routing: Four layers suffice", *Advances in Computing Research*, Ed. F. P. Preparata **2** (1984).
23. M. L. Brady and D. J. Brown, "Optimal multilayer channel routing with overlap", in *Advanced Research in VLSI*, Ed. C. E. Leiserson, The MIT Press (1986).
24. D. Braun, J. Burns, S. Devadas, H. K. Ma, K. Mayaram, F. Romeo and A. Sangiovanni-Vincentelli, "A new multi-layer channel router", *Proceedings of the 23rd Design Automation Conference* (1986).
25. P. Bruell and P. Sun, "A greedy three layer channel router", *Proceedings of the International Conference on Computer-Aided Design* (1985).
26. Y. K. Chen and M. L. Liu, "Three-layer channel routing", *IEEE Transactions on Computer-Aided Design* **3 (2)** (1984) 156–163.
27. J. Cong, D. F. Wong and C. L. Liu, "A new approach to three and four-layer channel routing", *IEEE Transactions on Computer-Aided Design* **7(10)** (1988) 1094–1104.
28. R. J. Enbody and H. C. Du, "Near optimal n-layer channel routing", *Proceedings of the 23rd Design Automation Conference* (1986) 708–714.
29. S. E. Hambrusch, "Channel routing algorithms for overlap models", *IEEE Transactions on Computer-Aided Design* **CAD-4(1)** (1985) 23–30.
30. S. E. Hambrusch, "Using overlap and minimizing contact points in channel routing", *Proceedings of the 21st Allerton Conference* (1983).
31. D. Zhou, "An optimal channel routing algorithm in the restricted wire overlap model", *INTEGRATION: the VLSI Journal* **9** (1990) 163–177.
32. M. Kaufmann, "Allowing overlaps makes switch-box layouts nice", *Proceedings of the* 2^{nd} *IEEE Symposium on Parallel and Distributed Processing* 1990 267–274.