A Near Optimal Algorithm for Routing Compatible Nets Inside a Rectangle‡

Teofilo F. Gonzalez
Department of Computer Science
The University of California
Santa Barbara, CA 93106

and

Si-Qing Zheng
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803

Abstract: The problem of routing compatible nets inside a rectangle (*RCNIR*) under the Manhattan wiring model is discussed. Our objective is to interconnect all the signal nets by introducing the least number of additional tracks and columns. We show that a two layer wiring with no more than three additional tracks and four additional columns than an optimal-area layout can be constructed in $O(n \log n)$ time, where n is the number of terminals.

Keywords: Algorithms, wire routing, VLSI design automation, switch-box routing, minimum area, near-optimal routing.

‡ This research was supported in part by the National Science Foundation under Grant DCR - 8503163.

I. Introduction

In a typical multi-phase VLSI layout system the layout problem is reduced to solving a collection of rectangle routing problem ([R]). In the *rectangle routing problem,* we are given a rectangle R and a set of *signal nets,* $N = \{n_1, n_2, ..., n_m\}$, each net consists of a set of terminals located on the boundary of R. The objective is to introduce wires inside R on k conducting layers in such a way that a set of design rules are satisfied, all the terminals from the same net are connected, and no two terminals from different nets are electrically common. The rectangle routing problem is also called the *switch-box routing problem.* A special case of this problem is called the *channel routing problem,* in which all terminals are located along two parallel lines. In this paper we study the rectangle routing problem under the *Manhattan wiring model.* In the Manhattan wiring model, a uniform grid is used and any two wires that have a grid point in common must cross each other. Two conducting layers are sufficient for Manhattan wiring model, one for running horizontal wires and the other for vertical wires.

The Manhattan channel routing problem, in which the objective is finding a wiring with minimum distance between the two shores (called *channel width*), is NP-complete ([Sz]). Considerable research effort has concentrated on developing good heuristics for this problem. Most of these algorithms do not guarantee provable good solutions (for more details about these algorithms the reader is referred to [BBL], [BP], [D], [RF], and [YK]). However, when all terminals located along two parallel lines are compatible, i.e., there is at most one terminal along any vertical line, the situation is different. This problem can be transformed to a routing problem in which all terminal points appear along one horizontal line. For this case, the $O(n \log n)$-time algorithm in [HS] generates an optimal wiring, where n is the number of terminals. This algorithm is best possible with respect to the time complexity bound [GLL], i.e, it is not possible to obtain a significantly faster algorithm if one restricts to the decision tree computation model. For the rectangle routing problem with terminals located along the four sides of rectangle R, only a few heuristics have been suggested (e.g. [Lu]). The combinatorial nature of this problem is still far from well understood.

Another well known wiring model is called the *knock-knee model.* As the Manhattan model, a uniform grid is used for running wires in the knock-knee model; the only difference is that two wire bends are allowed to share a common grid point. The main advantage of the knock-knee model is that it allows wirings that are more compact. For example, an optimal solution for the two-terminal-net channel routing problem can be obtained in $O(n \log n)$ time [PL]; and for the two-terminal-net rectangle routing problem a solution can be obtained in $O(n \log n)$ time, if it exists [K][MP]. Multiterminal-net channel routing problem and multiterminal-net rectangle routing problem are NP-complete [Sa]. Several approximation algorithms for these multiterminal-net routing problems have been designed [GZ][MP][MPS][GZ]. The main disadvantage of the knock-knee model is that even a simple routing solution may require more than two conducting layers [Li][BB]. It is not known how to generalize the knock-knee mode routing algorithms to obtain two-layer knock-knee free layouts with minimum area.

In this paper we study the problem of *Routing Compatible Nets Inside a Rectangle (RCNIR).* We present an algorithm that generates two-layer near-optimal wirings in $O(n \log n)$ time if every net contains two terminals, where n is the total number of terminals. Our algorithm implies a set of sufficient conditions for the existence of wiring solutions. Using this algorithm, other rectangle routing problems may be solved by paying a small penalty (with respect to the area). For example, the rectangle routing problem with neighborless terminals (to each side of every terminal there is an empty space) can be transformed into one with compatible nets by adding one additional track and one additional column. Furthermore, our algorithm can be used as an approximation algorithm for constructing Manhattan wirings for any two-terminal-net rectangle routing instance by inserting additional tracks and columns such that the routing area is no more than four times of the original rectangle. We believe that our algorithm will also route signal nets in more complex problems optimally or near-optimally.

II. Preliminaries

Let $I = (R, N)$ represent a problem instance, where R is a uniform rectangular *grid* formed by horizontal grid lines (called *tracks*) with y-coordinate values 0, 1, ..., h and vertical grid lines (called *columns*) with x-coordinate values 0, 1, ..., w. The horizontal grid lines with y-coordinate values 0 and h, are called the *bottom* and *top* boundary of R, respectively. The vertical lines with x-coordinate values 0 and w are called the *left* and *right* boundary of R, respectively. The bottom, top, left and right boundaries of R are labeled b, t, l and r, respectively. We use $T(1)$, $T(2)$, ..., $T(h-1)$ to refer to the tracks with y-coordinate values 1, ..., $h-1$, and we use $C(1)$, $C(2)$, ..., $C(w-1)$ to refer to the columns with x-coordinate values 1, 2, ..., $w-1$ in R. Let $N = \{n_1, n_2, ..., n_m\}$ be the collection of *nets*. Each net n_i consists of two *terminals* located on the boundary grid points of R (excluding the corners of R). We assume that the collection of nets is *compatible*, i.e., no two terminals are located along the same track or the same column. We say that a track or column is *empty* if there is no terminal located on its boundary points. Figure 2.1 gives a problem instance with $h = w = 11$ and $m = 10$.

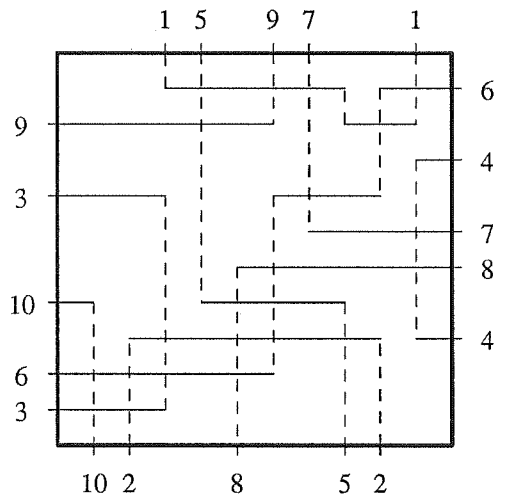

Figure 2.1: Problem instance.

A net is defined by the coordinates of its two terminals, i.e., $n_i = \{(x_1, y_1), (x_2, y_2)\}$, where (x_1, y_1) and (x_2, y_2) represent the (boundary) grid points where the two terminals of the net are located. The set N of net is partitioned into the following classes: N_{tt}, N_{bb}, N_{ll}, N_{rr}, N_{tb}, N_{lr}, N_{rt}, N_{rb}, N_{lt}, and N_{lb}, where N_{xy} for $x, y \in \{t, b, l, r\}$, represent the set of nets with one terminal located on side x and the other terminal located on side y. Net i in figure 2.1 belongs to the ith class of nets defined above. Classes of nets fall into one of the following groups depending on the location of their two terminals: *local* (terminals are located on the same side of R, i.e., N_{tt}, N_{bb}, N_{ll}, N_{rr}), *neighboring* (terminals are located on adjacent sides of R, i.e., N_{rt}, N_{rb}, N_{lt} N_{lb}), and *global* (terminals are located on opposite sides of R, i.e, N_{tb}, N_{lr}). Let $N_{neighbor} = N_{rt} \cup N_{rb} \cup N_{lt} \cup N_{lb}$. For a net $n_i \in N_{neighbor}$, $X(n_i)$ is defined to be the x-coordinate value of

its terminal located at the top or bottom boundary of R and $Y(n_i)$ is defined to be the y-coordinate value of its terminal located at the left or right boundary of R.

Under the Manhattan wiring model, a *wire layout* $W = \{W_1, W_2 ..., W_m\}$ is a set of edge-disjoint subgraphs of the grid R (not including its boundary) such that W_i connects the two terminals in net n_i. We say that a wire layout W has a *wire conflict* at grid point p if there exists W_i and W_j, $i \neq j$, such that they share the grid point p without crossing each other. We say that a wire layout W is *valid* under the Manhattan wiring model if there is no wire conflict among the wires in W. A valid wire layout W can be transformed into a two-layer *wiring* by assigning horizontal wire segments in one layer and vertical wire segments in the other layer, and introducing contact cuts (*vias*) whenever necessary. In the layout given in figure 2.1, the dashed lines are assigned to one layer and the solid lines are assigned to another layer.

The density of column i, denoted h_i, is the number of nets $\{(x_1, y_1), (x_2, y_2)\}$ such that $x_1 \leq i \leq x_2$, and the track density for row j, denoted by w_j, is the number of nets $\{(x_1, y_1), (x_2, y_2)\}$ such that $y_1 \leq j \leq y_2$. In figure 2.1, $h_3 = 4$, $h_5 = 6$, $w_2 = 5$, and $w_7 = 5$. Let $D_h(N) = \mathbf{max}\{h_i \mid 1 \leq i \leq w-1\}$ be the *column density* and let $D_w(N) = \mathbf{max}\{w_j \mid 1 \leq j \leq h-1\}$ be the *track density*. In figure 2.1, $D_h(N) = h_5 = 6$ and $D_w(N) = w_4 = 6$. Clearly, for $I = (R, N)$ if $D_h(N) > h$-1 or $D_w(N) > w$-1 it is impossible to wire the set of nets N inside R. Therefore, the conditions $D_h(N) < h$ and $D_w(N) < w$ are necessary conditions for a valid wiring to exist inside R. Figure 2.2 shows a problem instance that does not have a valid Manhattan wiring even though $D_h(N) \leq h$-1 and $D_w(N) \leq w$-1. Therefore, these conditions are not sufficient. These conditions can be combined into one, as suggested by the following lemma.

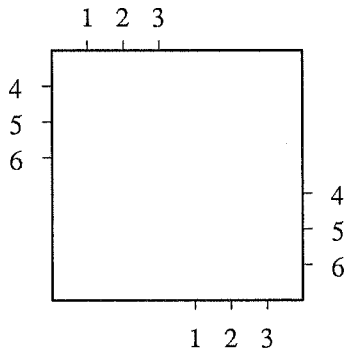

Figure 2.2: Problem without a wiring inside the given area.

Lemma 2.1: For problem instance $I = (R, N)$,

(i) if $w \geq h$ then $D_w(N) < w$;

(ii) if $h \geq w$ then $D_h(N) < h$; and

(iii) if $w = h$ then $D_w(N) < w$ and $D_h(N) < h$.

Proof: Since the proof of the three parts is similar, we only prove (i). The only nets that contribute to

$D_w(N)$ are the nets in $N_{tb}, N_{ll}, N_{rr}, N_{lr}, N_{lt}, N_{lb}, N_{rt}$ and N_{rb}. Each of these nets contributes at most one to $D_w(N)$. Therefore,

$$D_w(N) \leq |N_{tb}| + |N_{ll}| + |N_{rr}| + |N_{lr}| + |N_{lt}| + |N_{lb}| + |N_{rt}| + |N_{rb}|.$$

Since these nets consist of two terminals located on the boundary of R and the set of nets is compatible, it must be that

$$2|N_{tb}| + 2|N_{ll}| + 2|N_{rr}| + 2|N_{lr}| + 2|N_{lt}| + 2|N_{lb}| + 2|N_{rt}| + 2|N_{rb}| \leq h + w - 2 < 2w.$$

Therefore, $D_w(N) < w$.

□

We consider the problem of wiring $I = (R, N)$ in a rectangle $\bar{R}$ containing R with a set $\bar{N}$ of terminals on the boundaries of $\bar{R}$, where $\bar{N}$ is obtained by projecting the terminals in N to the boundaries of $\bar{R}$. Let $\bar{w}$ and $\bar{h}$ be the width and height of $\bar{R}$, and let $w^* = \mathbf{max}\{w, D_w(N)+1\}$ and $h^* = \mathbf{max}\{h, D_h(N)+1\}$. The values w^* and h^* are lower bounds for the width and height of $\bar{R}$, i.e., $\bar{w} \geq w^*$ and $\bar{h} \geq h^*$. Rectangle $\bar{R}$ is obtained by adding additional tracks and/or columns. In this paper we show that any problem instance can be wired inside a rectangle $\bar{R}$ with dimensions $\bar{w} \leq w^* + 3$ and $\bar{h} \leq h^* + 4$.

Our approach to the *RCNIR* problem consists of solving a sequence of increasingly more difficult subproblems. In section III we present a trivial algorithm for solving the *RCNIR* problem when there are only neighboring nets. Then we present an algorithm for the case when all the nets present are type N_{tt}, N_{bb} and N_{tb}. We then combine these two subproblems into one, which we call the *RRCNIR* problem, and present an algorithm for its solution. In section IV we use these algorithms to solve the *RCNIR* problem.

III. Restricted RCNIR Problem

In this section we solve increasingly more difficult subproblems of the *RCNIR* problem. We begin by discussing an algorithm to solve the *RCNIR* problem when there are only neighboring nets. Then we present an algorithm for the case when all the nets present are type N_{tt}, N_{bb} and N_{tb}. We then combine these two subproblems into one and present an algorithm for its solution. In the next section we use the algorithms developed in this section to solve the *RCNIR* problem. Throughout this section we assume that $w \geq w^*$ and $h \geq h^*$. In section IV we show how to deal with problems that do not satisfy this restriction.

The simpliest restricted *RCNIR* problem is one in which all nets are neighboring nets. This problem is trivial since one can obtain a wiring simply by connecting the two terminal in each net with an "L" shaped wire. Another special case is the one in which $N_{neighbor} \cup N_{ll} \cup N_{rr} \cup N_{lr} = \emptyset$, i.e., all nets are type N_{tt}, N_{bb} or N_{tb}. This version of the *RCNIR* problem can be transformed into a problem in which a wiring with the least number of tracks can be constructed in $O(n \log n)$ time ([HS] and [GLL]), where n is th total number of terminals. The following algorithm *ROUTE* 1 is similar to the ones given in [HS] and [GLL]. Algorithm *ROUTE* 1 partitions N into $m = D_h(N)$ subsets $N^1, N^2, ..., N^m$. Each set N^i is computed by procedure *LR_RUN*. Procedure *LR_RUN* constructs a subset $N' \subseteq N$ by a left-to-right scan of the

columns such that $D_h(N\text{-}N') = D_h(N) - 1$, $D_h(N') = 1$, and N' includes the net whose leftmost terminal is the leftmost terminal of nets in N. Algorithm *ROUTE* 1 and procedure *LR_RUN* are formally defined below.

algorithm *ROUTE* 1
  procedure *LR_RUN*(N)
    begin
      Let $N' \leftarrow \emptyset, N'' \leftarrow N$ and $j \leftarrow 0$;
      while M $\leftarrow$ { $\{(x_1, y_1), (x_2, y_2)\} \mid \{(x_1, y_1), (x_2, y_2)\} \in N''$ and **min**$\{x_1, x_2\} \geq$ j$\} \neq \emptyset$ **do**
        begin
          let n_k be the net in M whose leftmost terminal is closest to column j;
          $N' \leftarrow N' \cup \{n_k\}$;
          $N'' \leftarrow N''$ - $\{n_k\}$;
          let j be the column for the rightmost terminal of net n_k;
        end
      return(N');
    end
  end of *LR_RUN*;

  begin
    $i \leftarrow 0$;
    while $N \neq \emptyset$ **do**
      begin
        $i \leftarrow i + 1$;
        $N^i \leftarrow$ *LR_RUN* (N);
        $N \leftarrow N$ - N^i;
      end
    Route each net in N^i on the i th track;
  end
end of *ROUTE* 1;

Similarly, we define procedure *RL_RUN*(N) which constructs a subset $N' \subseteq$ N by a right-to-left scan of the columns such that $D_h(N\text{-}N') = D_h(N) - 1$, $D_h(N') = 1$ and N' includes the net whose rightmost terminal is the rightmost terminal of nets in N. Clearly, any sequence of executions of *LR_RUN*(N) and *RL_RUN*(N) guarantees a wiring with the same number of tracks.

In the remaining part of this section we consider the *RCNIR* problem in which $N_{ll} \cup N_{rr} \cup N_{lr} = \emptyset$ and $N_{neighbor} \cup N_{tt} \cup N_{bb} \cup N_{tb} \neq \emptyset$. We call this *RCNIR* problem the *restricted RCNIR problem (RRCNIR)*. Procedure *PART* defined below partitions N into $D_h(N)$ mutually disjoint subsets. Later on we show how to use this partition to solve the *RRCNIR* problem.

procedure *PART*(N);
  begin
    $m_0 \leftarrow m_1 \leftarrow m_2 \leftarrow 0$;
    while $N \neq \emptyset$ **do**
      begin
        $M \leftarrow LR\_RUN(N)$;
        Let i be the number of neighboring nets in M; /* clearly, $0 \leq i \leq 2$ */
        $m_i \leftarrow m_i + 1$;
        S(i,m_i) $\leftarrow M$;
        $N \leftarrow N - M$;
      end
  end
end of procedure *PART*;

Lemma 3.1: The sets $S(i,j)$, for $0 \leq i \leq 2$ and $1 \leq j \leq m_i$, constructed by procedure *PART* satisfy the following properties:

(i) there are exactly i neighboring nets in $S(i,j)$;

(ii) $D_h(S(i,j)) = 1$; and

(iii) $m_2 + e \geq m_0$, where e is the number of empty tracks in R.

Proof: Since the proof for (i) and (ii) is straight forward, we only prove (iii). Clearly, $h\text{-}1 = 2m_2 + m_1 + e$, $D_h(N) = m_0 + m_1 + m_2$, and $h > D_h(N)$. Substituting the first two equations in the last equation, we know that $m_2 + e \geq m_0$. This completes the proof of the lemma.

□

This partition allows us to reduce the *RRCNIR* problem to a simpler one by routing a subset of nets in N as follows.

procedure *ROUTE*2(R,N);

begin

invoke procedure *PART*(N) to obtain sets $S(i,j)$;

for each $S(1,j)$ route the set of nets $S(1,j)$ in track $T(Y(n_p))$, where n_p is the neighboring net in $S(1,j)$;

$l \leftarrow$ **min**$\{m_0, e\}$;

for each $S(0,j)$, m_0-l+1 $\leq j \leq m_0$, route the nets in $S(0,j)$ in an empty track;

for each $S(2,j)$, m_0-l+1 $\leq j \leq m_2$, route the nets in $S(2,j)$ in the two tracks $T(Y(n_p))$ and $T(Y(n_q))$, where n_p and n_q are the two neighboring nets in $S(2,j)$;

/* if $m_0 \leq e$ then all nets are routed, otherwise since $m_2 \geq m_0 - e$ (lemma 3.1 (iii)) the only remaining unrouted nets are in sets $S(0,j)$ and $S(2,j)$ for $1 \leq j \leq m_0 - l$. These sets will be referred to by $S'(0,j)$ for $1 \leq j \leq m_0'$ and $S'(2,j)$ for $1 \leq j \leq m_2'$ */

let $I' = (R', N')$ denote the new *RRCNIR* problem instance by deleting all used tracks and columns, and all routed nets;

end

end of procedure *ROUTE*2

Lemma 3.2: There are no wire conflicts in the wiring generated by procedure *ROUTE*2.

Proof: Since the proof is straight forward, it is omitted.

□

If just before executing procedure ROUTE2 it was the case that $m_0 \leq e$, then we have constructed a wiring for the original problem $I = (R, N)$. Otherwise we only need to solve the remaining *RRCNIR* problem $I' = (R', N')$. Let $S_0' = \{i \mid n_i \in S'(0,j) \text{ for } 1 \leq j \leq m_0'\}$.

Lemma 3.3: If $m_0 > e$, then $D_h(N') = h'$-1 = $2m_0' = 2m_2'$ and $D_h(S_0') = m_0' = m_2'$, where h' is the height of R'.

Proof: Since the proof is straight forward, it is omitted.

□

In what follows we give an algorithm that solves problem $I' = (R', N')$ by a sequence of m_2' steps. In each of these steps a set $S'(2,j)$ and a subset of S_0' will be selected and routed. We introduce additional notation to characterize the intermediate instance and to define the rules for the selection of the set $S'(2,j)$ and the subset of S_0' at each step.

Let a_{tl} and a_{tr} (a_{bl} and a_{br}) represent the x-coordinate value of the leftmost and the rightmost terminals of the nets in S'_0 located on the top (bottom) boundary of R' which are not yet connected. We use tl, bl, tr, and br to indicate conditions $a_{tl} < a_{bl}$, $a_{bl} < a_{tl}$, $a_{tr} > a_{br}$ and $a_{br} > a_{tr}$, respectively. We call these conditions *type I* conditions. Let

A_{tl} = **max**{Y(n_i) | n_i has not yet been connected, n_i is neighboring net in set $S'(2,j)$ and n_i has a terminal located at the left boundary of R'},

A_{tr} = **max**{$Y(n_i)$ | n_i has not yet been connected, n_i is a neighboring net in set $S'(2,j)$ and n_i has a terminal located at the right boundary of R'},

A_{bl} = **min**{$Y(n_i)$ | n_i has not yet been connected, n_i is a neighboring net in set $S'(2,j)$ and n_i has a terminal located at the left boundary of R'} and

A_{br} = **min**{$Y(n_i)$ | n_i has not yet been connected, n_i is a neighboring net in set S'(2,j) and n_i has a terminal located at the right boundary of R'}.

We use the notation TL, BL, TR and BR to indicate the condition $A_{tl} > A_{tr}$, $A_{bl} < A_{br}$, $A_{tl} < A_{tr}$, and $A_{bl} > A_{br}$, respectively. We call these conditions *type II* conditions. Hence, any intermediate instance can be characterized by a *condition pair*, X:Y, where condition X is of type I and condition Y is of type II. For example, tl:TL indicates that $a_{tl} < a_{bl}$ and $A_{tl} > A_{tr}$. There are 16 possible such condition pairs. However, only half of them are essential.

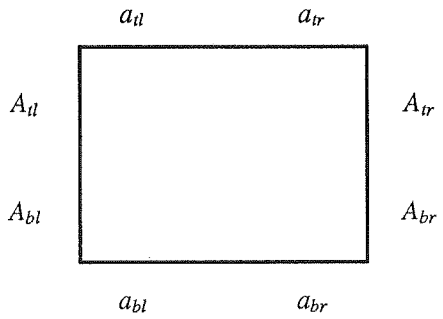

Figure 3.1: Notation.

Lemma 3.4: Problem instance $I' = (R', N')$ satisfies at least one of the condition pairs tl:TL, tr:TR, bl:BL, br:BR, tl:BL, tr:BR, bl:TL, and br:TR.

Proof: Suppose there is a problem instance that does not satisfy any of the eight condition pairs. Since either tl or bl holds, it cannot be that TL holds or BL holds (because it contradicts that either tl:TL, tl:BL, bl:TL, or bl:BL holds). Therefore, it must be that TR and BR hold. But then at least one of tr:TR, tr:BR, br:TR or br:BR holds since either tr or br holds. A contradiction.

□

We partition the eight condition pairs in lemma 3.4 into two groups. We call condition pairs $tl:TL$, $tr:TR$, $bl:BL$ and $br:BR$ as *type A condition pairs.* The remaining four condition pairs ($tl:BL$, $tr:BR$, $bl:TL$ and $br:TR$) are called *type B condition pairs.* These two types of condition pairs satisfy the following property.

Lemma 3.5: Let $I' = (R', N')$ be a problem instance that does not satisfy any type A condition pair. Then either

(i) $tl:BL$ and $br:TR$ hold, or

(ii) $tr:BR$ and $bl:TL$ hold.

Proof: From lemma 3.4 we know that at least one of the type B condition pairs holds. Assume it is condition pair $tl:BL$ (the proof for the other three cases is similar). To prove the lemma it is only necessary to show that $br:TR$ holds. Since tl holds, but $tl:TL$ does not hold (it is type A), it must be that TR holds. Since TR holds, but $tr:TR$ does not hold (it is type A), it must be that br holds. Therefore, $br:TR$ holds.

□

Our algorithm *ROUTE* 3 generates a wiring for the problem instance $I' = (R', N')$ with at most three additional tracks. Initially additional tracks will be added on the top side of R' (tracks t) and on the bottom side of R' (track b and s). When the algorithm terminates, we delete any of the three tracks (s, t and b) that were unused. Algorithm *ROUTE* 3 maintains the following invariants for track t (b):

(i) Track t (b) is located above (below) track $Y(n_i)$, where n_i is any not previously connected neighboring net.

(ii) Track t (b) may be used to route every net in S_0' which has not yet been routed, i.e., it is empty in the interval $[\mathbf{min}\{a_{tl}, a_{bl}\}, \mathbf{max}\{a_{tr}, a_{br}\}]$ (remember that the a's are defined with respect to unrouted nets in S_0').

Track s satisfies either the invariant for t or the invariant for b. Our algorithm *ROUTE* 3 iterates m_2' times. In each iteration we route all the nets in a set $S'(2,j)$ and a subset of the nets in S_0' that decreases the density of the nets in S_0' by one. The decision is based on which condition pair holds. In what follows we list the actions performed in each possible case.

case 1: Condition pairs $tl:TL$ or $tr:TR$ hold (figure 3.2).

Let j be such that the neighboring net $n_p \in S'(2,j)$ and $Y(n_p) = \mathbf{max}\{A_{tl}, A_{tr}\}$. Let n_q be the other neighboring net in $S'(2,j)$ and let $M' \leftarrow LR\_RUN(S'_0)$ $(RL\_RUN(S'_0))$ if $tl:TL$ $(tr:TR)$ holds. Route the nets in M' in track t, route the nets in $S'(2,j)$ - $\{n_p\}$ in track $T(Y(n_q))$ and route net n_p in track $T(Y(n_p))$ (and track $T(Y(n_q))$ if necessary). Remove $S'(2,j)$ and M' from further consideration. Since track $T(Y(n_p))$ satisfies the invariant for t, let the new track t be track $T(Y(n_p))$.

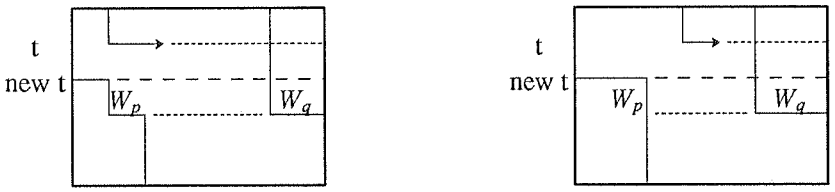

Figure 3.2: Example for case $tl:TL$ (the figure for $tr:TR$ is similar)

case 2: Condition pairs $bl:BL$ or $br:BR$ hold (figure 3.3).

Let j be such that neighboring net $n_p \in S'(2,j)$ and $Y(n_p) = \mathbf{min}\{A_{bl}, A_{br}\}$. Let n_q be the other neighboring net in $S'(2,j)$ and let $M' \leftarrow LR\_RUN(S'_0)$ $(RL\_RUN(S'_0))$ if $bl:BL$ $(br:BR)$ holds. Route the nets in M' in track b, route the nets in $S'(2,j)$ - $\{n_p\}$ in track $T(Y(n_q))$ and route net n_p in track $T(Y(n_p))$ (and track $T(Y(n_q))$ if necessary). Remove $S'(2,j)$ and M' from further consideration. Since track $T(Y(n_p))$ satisfies the invariant for b, let the new track b be track $T(Y(n_p))$.

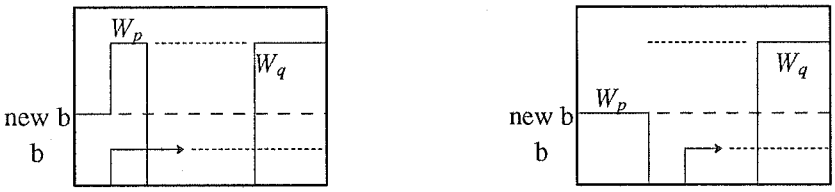

Figure 3.3: Example for case $bl:BL$ (the figure for $br:BR$ is similar).

case 3: Condition pairs $tl:BL$ and $br:TR$ hold and s satisfies the invariant for t (figure 3.4).
Let j be such that neighboring net $n_p \in S'(2,j)$ and $Y(n_p) = A_{bl}$. Let n_q be the other neighboring net in $S'(2,j)$ and let $M' \leftarrow LR\_RUN(S'_0)$. Route the nets in M' in track s, route the nets in $S'(2,j)$ - $\{n_p\}$ in track $T(Y(n_q))$ and route net n_p in track $T(Y(n_p))$ (and track $T(Y(n_q))$ if necessary). Remove $S'(2,j)$ and M' from further consideration. Since track $T(Y(n_p))$ satisfies the invariant for b, let the new track s be track $T(Y(n_p))$.

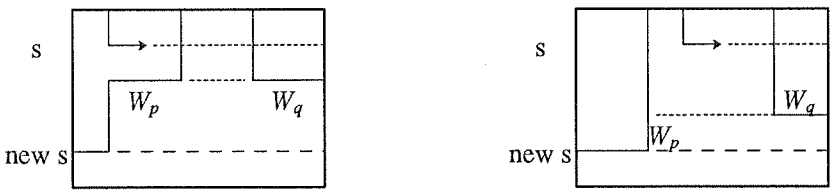

Figure 3.4: Example for case $tl:BL$ and $br:TR$.

case 4: Condition pairs $tl:BL$ and $br:TR$ hold, and s satisfies the invariant for b (figure 3.5).
Let j be such that the neighboring net $n_p \in S'(2,j)$ and $Y(n_p) = A_{tr}$. Let n_q be the other neighboring net in $S'(2,j)$ and let $M' \leftarrow RL\_RUN(S'_0)$. Route the nets in M' in track s, route the nets in $S'(2,j)$ - $\{n_p\}$ in track $T(Y(n_q))$ and route net n_p in track $T(Y(n_p))$ (and track $T(Y(n_q))$ if necessary). Remove $S'(2,j)$ and M' from further consideration. Since track $T(Y(n_p))$ satisfies the invariant for t, let the new track s be track $T(Y(n_p))$.

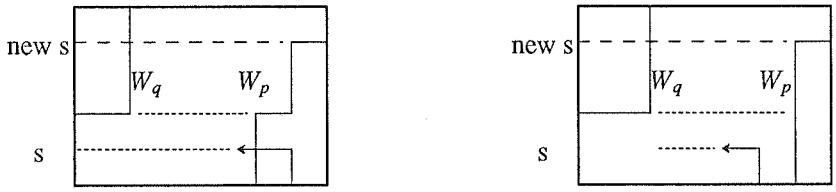

Figure 3.5: Example for case $tl:BL$, $br:TR$ and s satisfies the invariant for b.

case 5: Condition pairs $tr:BR$ and $bl:TL$ hold, and s satisfies the invariant for t (figure 3.6).
Let j be such that neighboring net $n_p \in S'(2,j)$ and $Y(n_p) = A_{br}$. Let n_q be the other neighboring net in $S'(2,j)$ and let $M' \leftarrow RL\_RUN(S'_0)$. Route the nets in M' in track s, route the nets in $S'(2,j)$ - $\{n_p\}$ in track $T(Y(n_q))$ and route net n_p in track $T(Y(n_p))$ (and track $T(Y(n_q))$ if necessary). Remove $S'(2,j)$ and M' from further consideration. Since track $T(Y(n_p))$ satisfies the invariant for b, let the new track s be track $T(Y(n_p))$.

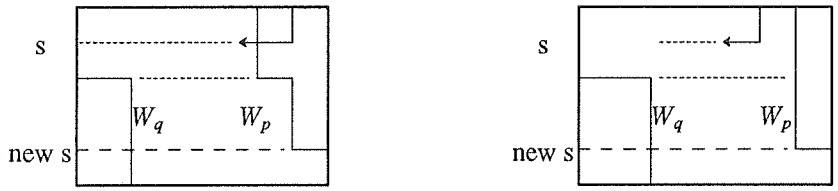

Figure 3.6: Example for case $tr:BR$, $bl:TL$, and s satisfies the invariant for t.

case 6: Condition pairs $tr:BR$ and $bl:TL$ hold, and s satisfies the invariant for b (figure 3.7).
Let j be such that the neighboring net $n_p \in S'(2,j)$ and $Y(n_p) = A_{tl}$. Let n_q be the other neighboring net in $S'(2,j)$ and let $M' \leftarrow LR\_RUN(S'_0)$. Route the nets in M' in track s, route the nets in $S'(2,j)$ - $\{n_p\}$ in track $T(Y(n_q))$ and route net n_p in track $T(Y(n_p))$ (and track $T(Y(n_q))$ if necessary). Remove $S'(2,j)$ and M' from further consideration. Since track $T(Y(n_p))$ satisfies the invariant for t, let the new track s be track $T(Y(n_p))$.

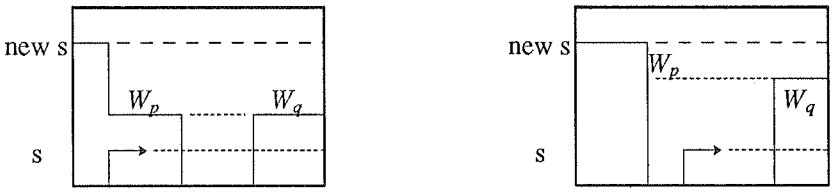

Figure 3.7: Example for case $tr:BR$, $bl:TL$, and s satisfies the invariant for b.

Figure 3.8 gives an example in which all nets are routed using the rules in procedure *ROUTE* 3. Initially s satisfied the invariant for b. The condition pairs satisfied at each step are: ($tl:TL$), ($bl:TL$, s is in the bottom), and ($tl:TL$).

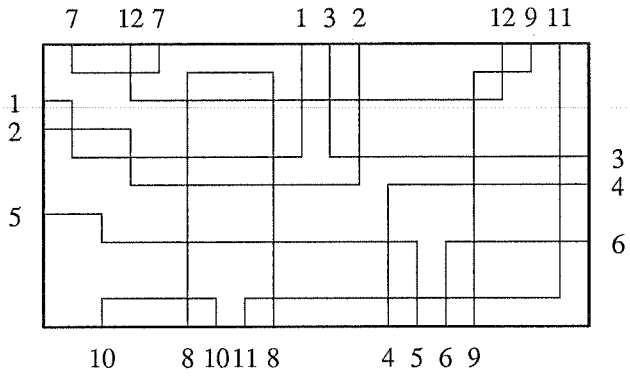

Figure 3.8: Example for *ROUTE* 3.

Lemma 3.6: Algorithm *ROUTE* 3 generates a wiring with at most three additional tracks for any problem instance $I' = (R', N')$ generated by procedures *ROUTE* 2.

Proof: By lemma 3.3, we know that $D_h(N') = h'\text{-}1 = 2m_2' = 2m_0' = 2D_h(S_0')$. In each iteration all the nets in one set $S'(2,j)$ and a subset of S_0' are connected. This selection is made depending on which of the condition pairs holds. By lemma 3.4 we know that for every problem instance at least one of eight condition pairs holds. After each iteration $D_h(S_0')$ decreases by one. Hence, algorithm *ROUTE* 3 terminates after m_2' iterations and all nets in N' are connected. To show that there are no wire conflicts among the wires generated by *ROUTE* 3 one needs to show that one of case 1-6 (above) holds (lemma 3.4 and 3.5) and that the routing performed in each of these cases introduces no conflicts. For brevity, we omit the proof of the latter fact.

□

Theorem 3.1: For any *RRCNIR* instance $I = (R, N)$ a wiring can be found in a rectangle R'' with height h'' $\leq h + 3$ and width $w'' = w$ in $O(n \log n)$ time, where n is the number of nets in N.

Proof: By lemmas 3.2, 3.3 and 3.6, we know that the combination of *ROUTE* 2 and *ROUTE* 3 solves the *RRCNIR* problem. These lemmas also show that for any *RRCNIR* instance $I = (R, N)$ algorithms *ROUTE* 2 and *ROUTE* 3 constructs a wiring in a rectangle R'' with height $h'' \leq h + 3$ and width $w'' = w$. Since the proof for the time complexity bound is simple, it will be omitted.

□

Theorem 3.2: For any *RRCNIR* instance $I = (R, N)$, the total number of vias in the wiring generated by our algorithm is no more than $1.5v^*$, where v^* is the least number of vias in any wiring of I.

Proof: Since any wire connecting a neighboring net needs at least one via and any wire connecting a non-neighboring net needs at least two vias, $v^* = |N_{neighbor}| + 2|N - N_{neighbor}|$ is a lower bound of the total

number of vias in any wiring for I. All wires introduced by procedure *ROUTE* 2 have the minimum number of vias. We only need to consider the wires generated by the procedure *ROUTE* 3. All the wires connecting non-neighboring nets generated by procedure *ROUTE* 3 have a minimum number of vias. From procedure *ROUTE* 3 we know that no more than half of the the wires connecting neighboring nets generated by the procedure *ROUTE* 3 contain three vias and no less than half of the the wires connecting neighboring nets generated by the procedure *ROUTE* 3 contain one via. For a problem instance $I' = (R', N')$, let v denote the total number of vias in the wiring generated by the procedure *ROUTE* 3, and let $\bar{v}$ denote the minimum number of vias in any wiring of I'. Clearly,

$$2|N'\text{-}N'_{neighbor}| + |N'_{neighbor}| \le \bar{v} \le v \le 2|N'\text{-}N'_{neighbor}| + 2|N'_{neighbor}|.$$

By lemma 3.3 we know that $D_h(N'\text{-}N'_{neighbor}) \ge |N'_{neighbor}|/2$, which implies that $|N'\text{-}N'_{neighbor}| \ge |N'_{neighbor}|/2$. Therefore,

$$v / \bar{v} \le (2|N'\text{-}N'_{neighbor}| + 2|N'_{neighbor}|) / (2|N'\text{-}N'_{neighbor}| + |N'_{neighbor}|)$$
$$= 1 + |N'_{neighbor}| / (2|N'\text{-}N'_{neighbor}| + |N'_{neighbor}|) \le 1.5.$$

This completes the proof of the theorem.

□

IV. The General RCNIR Problem.

Our procedure, *ROUTE* 0, constructs a routing solution for the *RCNIR* problem consists of the following steps: normalization (*NORM*), neighboring net initial routing (*NN_INIT_ROUTE*), pre-routing (*PRE_ROUTE*), knock-knee elimination (*KK_ELIM*), remaining problem definition (*RP_DEF*), trivial net elimination (*TN_ELIM*), and routing the *RRCNIR* problem. Before formally defining these procedures, we briefly describe them below.

Basic Steps

Our problem instance is defined by $I = (R, N)$. Let w and h be the width and height of R. Let w^* and h^* be the lower bounds defined in section II. If $h < h^*$ (w < w^*) then add enough columns and tracks until $h = h^*$ (w = w^*). As we have shown before (figure 2.2) $h = h^*$ and $w = w^*$ are not sufficient conditions for a wiring to exist. Because of this, enough tracks and/or columns must be added to ensure wirability. We say that an instance $I = (R, N)$ is *normal* if $h \le w$, $h \ge h^*$, $w \ge w^*$, and at least one of tracks $T(1)$ or $T(h-1)$ and at least one of columns $C(1)$ or $C(w-1)$ is empty, i.e., does not contain any terminal. In the normalization step (*NORM*), problem instance I is made a normal instance by adding tracks and columns. At the end of this step $h \ge h^* + 1$ and $w \ge w^* + 1$.

In the neighboring net initial routing step (*NN_INIT_ROUTE*) we construct a problem instance $I' = (R', N')$ for I by ignoring all the neighboring nets. These nets are actually routed by

"L" shaped wires. Thus, by eliminating the neighboring nets we mean deleting all tracks and columns in which there is a terminal from some neighboring net. Note that since every track and column has at most one terminal in it and every neighboring net contains a terminal in a track and in a column, it must be that the height (h') of R' is less than or equal to the width (w') of R' (remember that after normalization step h is less than or equal to w). We use N'_{xy} to denote the set of nets with one terminal located on boundary x of R' and the other located on boundary y of R'.

In the pre-routing step (*PRE_ROUTE*), we route a subset of nets that includes all nets in $N'_{ll} \cup N'_{rr} \cup N'_{lr}$. The wiring may include some conflicts. These conflicts are called knock-knees, i.e., two wires that bend at a grid point (see figure 4.8). In the knock-knee elimination step (*KK_ELIM*), we obtain a valid routing by modifying the previously constructed routing in such a way that all knock-knees are eliminated. In this step, at most two additional columns are introduced. After this step we have a valid routing, H, for a subset of nets. If all nets have been routed, the algorithm stops.

In the *RP_DEF* step we define the remaining problem, $I'' = (R'', N'')$, that needs to be solved. Some of the neighboring nets in I will be re-routed. We begin this process by defining a rectangular window, R'', in R' and the terminals in R' are projected to R'' (whenever possible). All nets with at least one terminal not located on the boundary of the window are routed on the outside of the window in the same tracks and columns they were routed in H, or if the net is a neighboring nets in I it is routed outside the window in the obvious way. As a result of this operation, we have a restricted version of the *RCNIR* problem studied in the previous section, with the possible exception that there could be some trivial global nets in set N''_{lr} (a trivial net is a net whose two terminals occupy the same track). These nets can be eliminated by routing them in the obvious way and deleting the track where they are located. Note that each time we delete one of these nets, we decrease h'' and D''_h by one. Problem instance I'' is routed by algorithms *ROUTE* 2 and *ROUTE* 3 given in the previous section. The final wiring inside the window is obtained by *ROUTE* 2 and *ROUTE* 3, and the one outside the window is obtained from the previous wirings outside R''. Remember that *ROUTE* 3 adds three tracks. Therefore, our procedure *ROUTE* 0 routes all nets inside a rectangle with area $h^* + 4$ and $w^* + 3$.

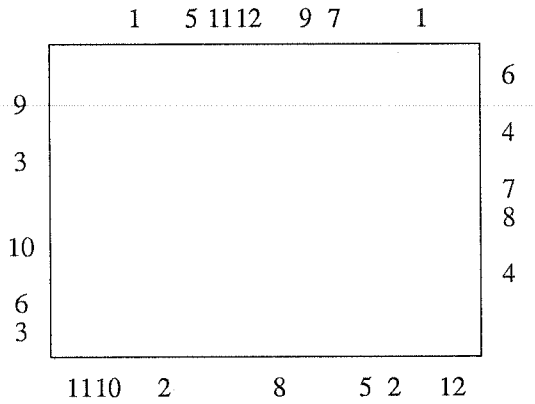

Figure 4.1: Problem instance $I = (R, N)$.

Now we explain in detail each of the steps in algorithm *ROUTE*0 when invoked with problem instance $I = (R, N)$. To clarify our presentation we will use the problem instance given in figure 4.1 as an example. Procedure *NORM* begins by rotating the rectangle 90 degrees if $w < h$. Then it adds at most one column to the left of R and at most $\mathbf{max}\{1, h - h^* + 1\}$ tracks to the top of R. Clearly $h \leq h^* + 1$ and $w \leq w^* + 1$. Figure 4.2 shows our example after procedure *NORM* introduces an empty track and column.

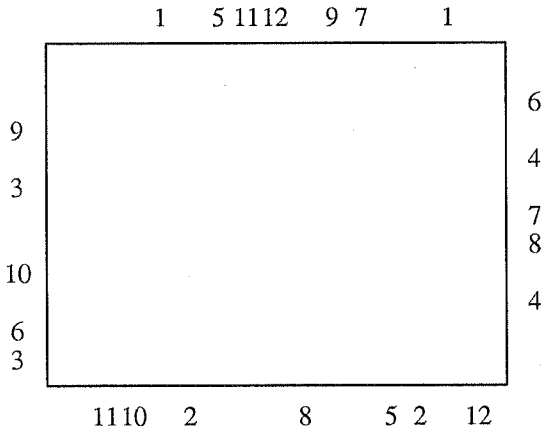

Figure 4.2: Problem instance after procedure *NORM*.

In the neighboring net initial routing step, procedure *NN_INIT_ROUTE* constructs a problem instance $I' = (R', N')$ from I by ignoring the neighboring nets. These nets are routed by an "L" shaped wire. Thus, we may delete all tracks and columns in which there is a terminal from some neighboring net. Note that since every track and column has at most one terminal in it and every neighboring net contains a terminal in a track and in a column, it must be that the height (h') of R' is less than or equal to the width (w') of R'. We use N'_{xy} to denote the set of nets with one terminal located on boundary x of R' and the other located on boundary y of R'. Figure 4.3 shows $I' = (R', N')$ generated for our example.

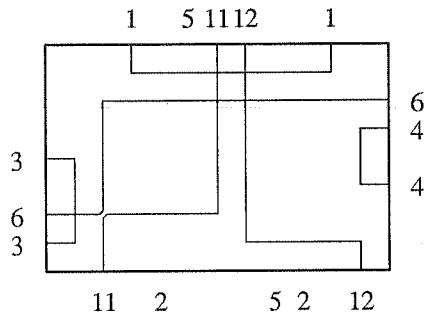

Figure 4.3: Problem instance $I' = (R', N')$.

The wires are introduced in phase one of procedure *PRE_ROUTE*

In the pre-routing step carried out by procedure *PRE_ROUTE* , we route a set of nets that includes all nets in $N'_{ll} \cup N'_{rr} \cup N'_{lr}$. Remember that I' has been normalized. Procedure *PRE_ROUTE* consists of two phases. In the first phase the algorithm iterates until either all nets in $N'_{ll} \cup N'_{rr} \cup N'_{lr}$ or all nets in $N'_{tt} \cup N'_{bb} \cup N'_{tb}$ have been routed. In the second phase the algorithm routes all the remaining nets (if any) in $N'_{ll} \cup N'_{rr} \cup N'_{lr}$. Let us now formally define each phase. In the first phase, we define a window G of R' whose size decreases at each iteration. Initially G is R'. Since I' is normal, then either the leftmost or rightmost column and, either the bottommost or topmost track must be empty. Let *left (right)* represent the x-coordinate value of the left (right) boundary of G and let *top (bottom)* represent the y-coordinate value of the top (bottom) boundary of G. Each column and track in G will be labeled either *available* or *unavailable* depending on whether we allow any further routing inside of G on it or not. Initially *all* tracks and columns are labeled available. Each available track or column has at most one terminal in it. If there are none, we say the track or column is *empty*. Note that initially all empty tracks and columns are labeled available. We use *ca (ta)* to represent the number of available columns (tracks) in G. From the initial conditions we know that $ta = h' - 1 \leq w' - 1 = ca$. At each iteration the algorithm decreases *ta* and *ca* by exactly two. Remember that the algorithm stops when either all nets in $N'_{ll} \cup N'_{rr} \cup N'_{lr}$ or all nets in $N'_{tt} \cup N'_{bb} \cup N'_{tb}$ have been routed. Initially all terminals from the nets in $N'_{ll} \cup N'_{rr} \cup N'_{lr} \cup N'_{tt} \cup N'_{bb} \cup N'_{tb}$ belong to available tracks or columns. Procedure *PRE_ROUTE* maintains the following invariant at each iteration.

(i) Either the topmost or bottommost column, and either the leftmost or rightmost column is empty.

(ii) All nets with at least one terminal located on the left or right side of R' between tracks 0 and *bottom*, and *top* and h'; and all nets with at least one terminal located on the top or bottom side of R' between columns 0 and *left*, and *right* and w', have been routed in $R' - G$ (i.e., the area in R' outside window G).

(iii) All terminals from the nets which have not yet been routed in $N'_{ll} \cup N'_{rr} \cup N'_{lr} \cup N'_{tt} \cup N'_{bb} \cup N'_{tb}$ belong to available tracks or columns, and have been projected to the boundary of G.

(iv) $2 \leq ta \leq ca$.

During each iteration in phase one of procedure *PRE_ROUTE* we route exactly two nets. Then we decrease G by one track and one column. The terminals from all unrouted nets in I' are projected to the new boundary of G. At the end of an iteration, we reduce G by eliminating all unavailable tracks (columns) adjacent to either the top (left) or bottom (right) boundary of G. Again, the terminals from all unrouted nets in I' are projected to the new boundary of G. Obviously, G decreases in size when this operation is performed. In what follows we explain what operations are performed at the beginning of each iteration in phase one.

Assume without loss of generality that the topmost track and the leftmost column in G are empty. (If this is not the case, G can be rotated 90, 180 or 270 degrees). Let x (y) be the rightmost (bottommost) column (track) in G where a terminal is located. It is important to note that a track at this point could be a track or a column in the original problem instance because it could have been rotated. There are four cases that need to be considered depending on the location of the terminals in column x and track y.

case 1: The terminal in column (track) x (y) is located on the bottom (right) side of G.
Figure 4.4 shows the actions performed in this case. Our shorthand notation is defined as follows. A dotted line indicates that the corresponding track or column is marked unavailable at this step; the dashed lines indicate an empty track or column; a solid line indicates a net routed at this step; and the shaded region indicates the area deleted at the end of this step. Note that in this case a knock-knee may be introduced. Also, we are not showing in the figure the complete wire introduced for the two nets (the missing part of the wire will only occupy part of the dashed track or column).

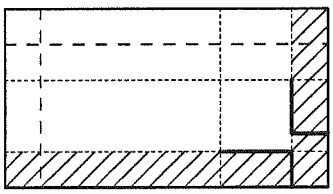

Figure 4.4: Case 1.

case 2: The terminal in column (track) x (y) is located on the bottom (left) side of G. Figure 4.5 shows the actions performed in this case.

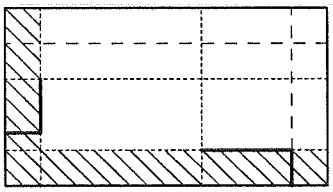

Figure 4.5: Case 2.

case 3: The terminal in column (track) x (y) is located on the top (right) side of G. Figure 4.6 shows the actions performed in this case.

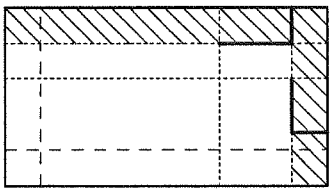

Figure 4.6: Case 3.

case 4: The terminal in column (track) x (y) is located on the top (left) side of G. Figure 4.7 shows the actions performed in this case.

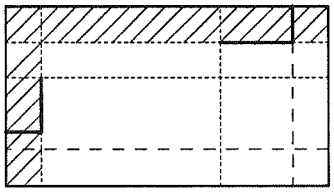

Figure 4.7: Case 4.

Since at each step one net in $N_{ll}' \cup N_{rr}' \cup N_{lr}'$ and one net in $N_{tt}' \cup N_{bb}' \cup N_{tb}'$ are routed, the first phase of *PRE_ROUTE* terminates when either all nets in $N_{ll}' \cup N_{rr}' \cup N_{lr}'$ or all nets in $N_{tt}' \cup N_{bb}' \cup N_{tb}'$ are routed. It is simple to verify that there may be wire conflicts (knock-knees). Figure 4.3 shows the type of wires introduced by phase one of procedure *PRE_ROUTE*.

In the second phase of procedure *PRE_ROUTE* we route all the remaining nets in $N_{ll}' \cup N_{rr}' \cup N_{lr}'$. This phase will do nothing if all of these nets have been routed. Otherwise, we know from phase one that all nets in $N_{tt}' \cup N_{bb}' \cup N_{tb}'$ have been routed. Since at each step ta and ca are decreased by two, we know that at the beginning of phase two $ta \leq ca$. Since each terminal is located on a separate track the number of unrouted nets in $N_{ll}' \cup N_{rr}' \cup N_{lr}'$ is at most $ta/2$. Therefore we route each remaining net in its own track

and in one available column (remember that $ta \leq ca$). The rectangle G is redefined as a single grid point inside the previous G. Phase two of procedure *PRE_ROUTE* will not affect the wiring in figure 4.3.

It is easy to verify that there may be wire conflicts. However, all of these conflicts are introduced in phase one of *PRE_ROUTE* and are of the following form. A bend of a wire connecting a net from $N'_{tt} \cup N'_{bb} \cup N'_{tb}$ shares a grid point with a bend of a wire connecting a net from $N'_{ll} \cup N'_{rr} \cup N'_{lr}$. A wire conflict of this form is called a *knock-knee*. The two different types, *type-1* and *type-2*, of knock-knees are shown in figure 4.8.

type-1 type-2

Figure 4.8: Knock-knees.

Lemma 4.1: For any normal instance $I' = (R', N')$, procedure *PRE_ROUTE* terminates with a partial wiring in R' such that

(i) all nets in $N'_{ll} \cup N'_{rr} \cup N'_{lr}$ are routed;

(ii) the wire conflicts, if any, among the wires introduced are knock-knees;

(iii) for any two wires forming a knock-knee, one is connecting a net in $N'_{ll} \cup N'_{rr} \cup N'_{lr}$ and the other is connecting a net in $N'_{tt} \cup N'_{bb} \cup N'_{tb}$; and

(iv) If there is a type-1 knock-knee at grid point $p = (x_p, y_p)$, $x_p \leq left$ ($x_p \geq right$), then all the vertical wires located in column x_q, $x_p < x_q \leq left$ ($right \leq x_q < x_p$), with their top (bottom) end point in track $y_q \leq y_p$ ($y_q \geq y_p$) [see figure 4.9a]. If there is a type-2 knock-knee at grid point $p = (x_p, y_p)$, $x_p \leq left$ ($x_p \geq right$), then all the vertical wires located in column x_q, $x_p < x_q \leq left$ ($right \leq x_q < x_p$), with their bottom (top) end point in track $y_q \geq y_p$ ($y_q \leq y_p$) [see figure 4.9b].

Proof: Since the proof is straight forward, it will be omitted.

□

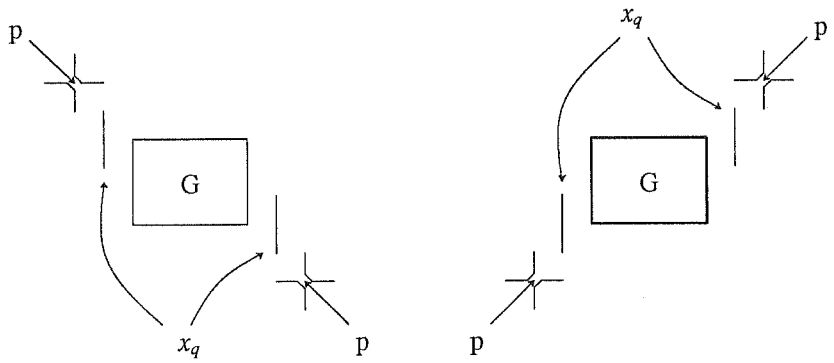

Figure 4.9: Problem instance $I = (R, N)$.

Procedure *KK_ELIM* adds at most two additional columns, one to the left of R' and the other to the right of R', to resolve all wire conflicts among the wires generated by procedure *PRE_ROUTE*. The operations performed are as follows. Shift all vertical wires connecting nets in $N_{ll} \cup N_{lr}$ in column $C(i)$, to column $C(i-1)$, for $1 \leq i \leq left$, and shift all vertical wires connecting nets in $N_{rr} \cup N_{lr}$ in column $C(i)$, to column $C(i+1)$, for $right \leq i \leq w-1$. By lemma 4.1, we know that the resulting routing has no conflicts. In figure 4.10 we show our example after procedure *KK_ELIM*.

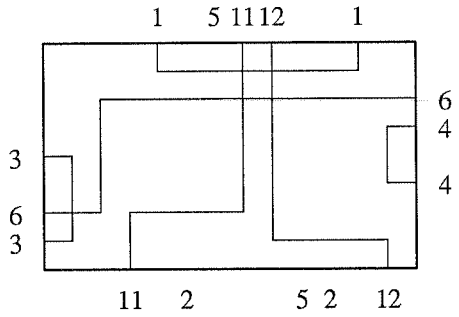

Figure 4.10: Our example after procedure *KK_ELIM*.

The remaining problem, $I'' = (R'', N'')$, is now defined by procedure *RP_DEF*. The window, R'', in R includes all tracks in R and only the columns from point *left* in G to point *right* in G. All nets with at least one terminal located outside the window R'' are routed on the outside of the window in the same tracks and columns they were routed just after procedure *KK_ELIM* or if the net is a neighboring nets in I, it is routed outside the window in the obvious way. The remaining terminals are projected to the boundary of R''. Figure 4.11 shows I'' for our example. As a result of this operation, we have a restricted version of the *RCNIR* problem studied in the previous section, with the possible exception that there could some be trivial global nets in set N''_{tr} (a trivial net is a net whose two terminals occupy the same track). These nets can be eliminated by routing them in the obvious way and deleting the track where they are located. Figure 4.12 shows our example after deleting trivial nets. Note that each time we delete one of these nets, h'' and D''_h decrease by one. Problem I'' is routed by algorithms *ROUTE* 2 and *ROUTE* 3 given in the previous section. Figure 4.11 shows the wiring for I'' in our example. The final routing is the wiring obtained by *ROUTE* 2 and *ROUTE* 3 plus all the previous routings outside R''. Figure 4.13 shows the final wiring. Remember that procedure *ROUTE* 3 adds at most three tracks. Our main theorem is given below.

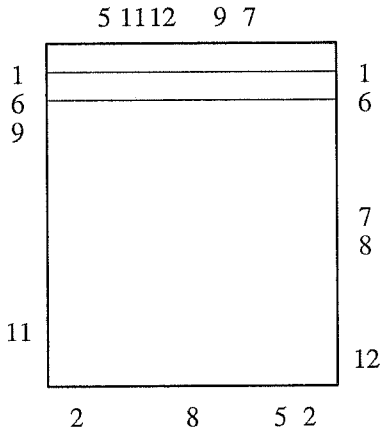

Figure 4.11: Problem instance I''.

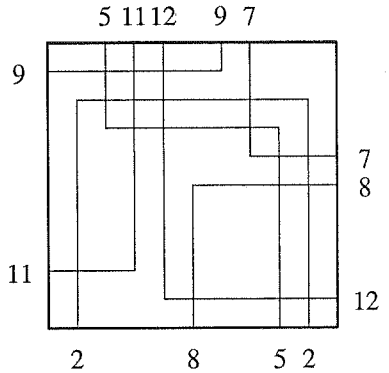

Figure 4.12: Problem instance I'' after deleting trivial nets. The wiring was generated by procedures ROUTE2 and ROUTE3.

Theorem 4.1: Procedure *ROUTE* 0 constructs for any *RCNIR* instance $I = (R, N)$ a wiring of no more than $1.5v^*$ vias, where v^* is the minimum number of vias in any wiring of I. The wiring has height $h^* + 4$ and width $w^* + 3$. The worst case time complexity for procedure *ROUTE* 0 is $O(n \log n)$ time, where n is the number of nets in N.

Proof: The only procedure that introduces additional vias is *ROUTE* 2. By theorem 3.2, we know that the total number of vias in no more than 1.5 times the optimal number of vias.

□

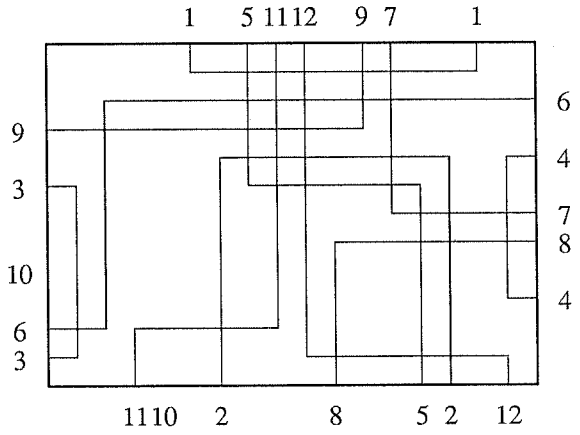

Figure 4.13: Final wiring for our example.

V. Discussion

We presented an algorithm for routing inside a rectangle compatible nets near-optimally. In a multi-phase *VLSI* layout system, the layout problem can be reduced to a collection of *RCNIR* problems and then a layout for each of these *RCNIR* problem can be constructed by our algorithm. The compatibility of the nets can be enforced by the global routing algorithm. Our algorithm can also be used as an approximation algorithm for routing arbitrary two-terminal-net Manhattan instances. Consider the problem instance shown in figure 5.1(a). In this instance, nets are not compatible. By introducing a row (column) between every two adjacent rows (columns) of R, and shift terminals on the top (left) boundary side of the resulting rectangle R' one unit distance to the left (downward), we can construct an RCNIR instance shown in figure 5.1(b). If $D_h(\mathrm{N}) < h$ and $D_w < w$, where h and w are the height and width of R, respectively, then $D_h(N') < h'$ and $D_w < w'$, where h' and w' are the height and width of R', respectively, and N' is the set of nets obtained from N by transformation described above. Applying our algorithm, we can construct a near-optimal Manhattan wiring for the newly constructed instance. Since $h' \leq 2h$ and $w' \leq 2w$, the area of the R' is about four times of the area of R.

(a) A non-*RCNIR* routing instance. (b) An *RCNIR* instance.

Figure 5.1: Transforming a non-*RCNIR* instance into an *RCNIR* instance.

As mention in section I, the Manhattan routing problem is more difficult than the knock-knee mode routing problem. The difficulty is caused by knock-knee free requirement. The two-terminal-net channel routing problem and rectangle routing problem under the knock-knee model can be formalized as multi-commodity flow problems which can be solved efficiently [F][PL][MP]. In contrast, these problems under the Manhattan model are NP-complete. The combinatorial nature of Manhattan routing problems is still far from well understood. Our algorithm, although dealing with a rather restricted version of the general Manhattan routing problems, is one step further towards better understanding of the general routing problems. We would like to point out two aspects of Manhattan rectangle routing problem. First, a non-trivial routable problem instance $I = (R, N)$ tend to be sparse, i.e. if there exists a wire layout for I then the terminals of N tend to be not so densely located along the boundary of R. Second, the Manhattan wire layouts tend to be not as compact as the knock-knee mode layouts, i.e in a Manhattan wire layout there are always many grid line segments not utilized unless the problem is trivial. These two aspects of non-trivial Manhattan routing problems result from the knock-knee free requirement. In spite of these weaknesses, Manhattan wiring model does have the advantage that it requires only two conducting layers. To improve the performance of our algorithm, one may consider to add an additional compaction phase which "squeezes" the layout constructed by our algorithm to obtain a more compact layout which satisfies the knock-knee free requirement. Also, one may consider to decompose a given routing problem into several relatively simpler subproblems, and use our algorithm to solve some of these subproblems.

To our knowledge, our algorithm is the first provably good algorithm for the Manhattan rectangle routing problem. Our algorithm implies a set of sufficient conditions for the existence of wiring layouts for the *RCNIR* problem. An interesting open problem is finding a set of necessary and sufficient conditions for the existence of wirings for the *RCNIR* problem. If nets consist of more than two terminals, the compatible-net rectangle routing problem becomes much more complex. There is no known efficient algorithm to solve this more general problem. The two-shore compatible net routing problem has many applications in solving a wide variety of routing problems, including our problem. We believe that many routing problems with sparse terminals can be transformed into compatible-terminal routing problem after paying a small penalty in terms of the number of additional tracks or columns.

VI. References

[BB] Brady, M.L. and D.J. Brown, "VLSI Routing: Four Layers Suffice", *Advances in Computing Research*, vol. 2, 1984.

[BBL] Baker, B. S., S. N. Bhatt and F. T. Leighton, "An Approximation Algorithm for Manhattan Routing", Proc. 15th ACM Symp. on Theory of Computing, 1983.

[BP] Burstein, M. and P. Pelavin, "Hierarchical Wire Routing," *IEEE Transactions on CAD*, vol. CAD-2, no. 4, 1983.

[D] Deutsch, D. N., "A Dogleg Channel Router," Proc. 13th Design Automation Conference, 1976.

[F] Frank, A., "Disjoint Paths in Rectilinear Grid", *Combinatorica*, 2(4), 1982.

[GK] Gao, S. and M. Kaufmann, "Channel Routing of Multiterminal Nets", Proceedings of the 28th Symposium on Foundations of Computer Science, 1987.

[GLL] Gupta, U. I., D. T. Lee and Y-T. Leung, "An Optimal Solution for the Channel-Assignment Problem", *IEEE Transactions on Computers*, vol. C-28, 1979.

[GZ] Gonzalez, T. and S.-Q. Zheng, "Grid Stretching Algorithms for Routing Multiterminal Nets Through a Rectangle", 1989.

[HS] Hashimoto, A., and J. Stevens, "Wiring Routing by Optimizing Channel Assignment within Large Apertures," Proc. 8th Design Automaton Conference, 1971.

[Li] Lipski, W. Jr, "An NP-Complete Problem Related to Three-Layer Channel Routing", Advances in Computing Research, vol. 2, 1984.

[Lu] Luk, W. K., "A Greedy Switch-Box Router", *INTEGRATION, the VLSI journal*, 3, 1985.

[MP] Mehlhorn, K. and F. Preparata, "Routing Through a Rectangle," *Journal of ACM*, vol. 33, no. 1, 1986, pp. 60-85.

[MPS] Mehlhorn, K., F. Preparata, and M. Sarrafzadeh, "Channel Routing in Knock-Knee Mode: Simplified Algorithms and Proofs", *Algorithmica*, no. 1, 1986.

[PL] F. Preparata and W. Lipski, Jr, "Optimal Three-Layer Channel Routing", *IEEE Transactions on Computer*, vol. 33, no. 5, 1984.

[R] Rivest, R. L., "The PI (placement and interconnect) system," Proc. 19th Annual Design Automation Conference, 1982.

[RBM] Rivest, R. L., 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.

[RF] Rivest, R. L. and C. M. Fiduccia, "A Greedy Channel Router," Proc. 19th Design Automation Conference, 1982.

[Sa] Sarrafzadeh, M., "Channel Routing Problem in Knock-Knee Mode is NP-Complete", *IEEE Transactions on CAD*, vol. CAD-6, no. 4, 1987.

[Sz] Szymanski, T. G., "Dogleg Channel Routing is NP-complete," *IEEE Transactions on CAD*, 1982.

[SR] Soukup, J., and J. C. Royle, "On Hierarchical Routing," *J. Digital Systems*, vol. 5, no. 3, 1981.

[U] Ullman, J. D., *Computational Aspects of VLSI*, Computer Science Press, 1984.

[YK] Yoshimura, T. and E. Kuh, "Efficient Algorithms for Channel Routing," *IEEE Transactions on CAD*, vol. CAD-1, no. 1, 1982.