PERGAMON

Computers and Electrical Engineering 25 (1999) 57–75

Computers and Electrical Engineering

Designing a channel router by hybrid methodology of top routing and bottom routing

J.-T. Yan*

Computer Systems Research Center, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan

Abstract

In this paper, based on the hybrid methodology of top routing and bottom routing, we propose an $O(N_{\mathrm{col}})$ approach for the channel routing problem, where N_{col} is the number of columns in a channel. Basically, top (bottom) routing is a track-assignment-based routing approach in a channel, i.e. a channel is routed track by track from top (bottom) to bottom (top) by running top (bottom) routing. In the proposed routing approach, the routing process is divided into two phases: *iterative-construction* phase and *merging-improvement* phase. In the iterative-construction phase, the net interval of each routing net is split into horizontal segments and these segments are further assigned track by track in a top-down or bottom-up manner. In the merging-improvement phase, the routing result is further improved by merging shorter segments in different tracks into longer segments for the reduction of the total wire length and the number of vias. Finally, the proposed approach has tested many published channels and the routing results are in the optimal number of tracks. For example, the Deutsch's difficult channel is routed in 19 tracks with automatic introduction of doglegs. In addition to the optimality of the number of tracks, the proposed approach obtains fewer vias and shorter total wire length than all other Manhattan channel routers. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Channel routing; Manhattan routing model; Track assignment; Top routing; Bottom routing; Deutsch's difficult channel

1. Introduction

In recent years, the advances in VLSI technology allow a highly complex system to be implemented on one single chip. In order to reduce fabrication time and cost, many computer-aided-design (CAD) tools have been developed to design such a complex VLSI system.

* E-mail: yan@vlsi,cis.nctu.edu.tw.

Generally speaking, the routing area usually occupies more than half of the final layout area in a typical VLSI circuit design. Hence, the routing process plays an important and time-consuming role in VLSI design automation. Basically, the routing process is divided into *global routing* and *detailed routing*. In global routing, each routing net in a circuit net-list is represented by a topological wiring according to the constraints of routing congestion and area. Furthermore, all of the routing nets in this net-list are routed and the routing area is minimized in detailed routing. If the routing area is minimized in detailed routing, the final layout area will be minimized in the routing process. Hence, it is necessary for the minimization of the final layout area to develop an optimal routing tool. Unfortunately, most of the routing problems are NP-complete and it seems that there is no optimal router to minimize the routing area in a chip. Hence, it is important for VLSI design automation to develop an efficient detailed routing tool.

In general, the objective of detailed routing is to accomplish all of the specified interconnections among circuit modules with smaller area. Among several available detailed routing strategies, channel routing always guarantees a fully completed result as the channel width and length are adjustable. Hence, channel routing is most often used in detailed routing, and the main goal of designing a channel router is to minimize the routing area by minimizing the number of tracks to be used.

In a two-layer non-overlap Manhattan grid model, a typical channel routing algorithm called left-edge was firstly proposed by Hashimoto and Stevens [1] in 1971. Soon after that, a modified version of the left edge algorithm was implemented in the Bell Labs' Polycell Layout System, LTX [2]. Basically, these algorithms did not allow the interval of any routing net to be split by the introduction of doglegs so that they could not reduce wasted routing area efficiently and, moreover, they used more tracks to solve a channel routing problem. In 1980, LaPaugh [3] further showed the NP-completeness of this channel routing problem without the introduction of doglegs. The term 'dogleg', where the interval of a routing net is split into two or more horizontal segments on different tracks and the Deutsch's difficult channel were introduced by Deutsch [4] in 1976. The doglegging technique was usually efficient for reducing the number of tracks in a channel and for breaking the cycles in a vertical constraint graph. The first doglegging router [4] was proposed and yielded a routing solution with 21 tracks for the Deutsch's difficult channel. Furthermore, the doglegging technique was combined with the net merging operation proposed by Yoshimura et al. [5] to yield a routing result with 20 tracks for the Deutsch's difficult channel. Besides that, the greedy router [6] extended the doglegging idea to permit the doglegging technique in any column that not necessarily contains any terminal pin of the doglegged net. Basically, this router scanned a channel in a left-to-right column-by-column manner, and optimized the utilization of all the wiring tracks in a greedy manner within a given column before proceeding to the next. As it is applied to route the Deutsch's difficult channel, this greedy router yielded a routing result with 20 tracks. Nevertheless, these routers did not yield an optimal routing result for the Deutsch's difficult channel. In contrast to these greedy routers, a hierarchical route [7] used the divide-and-conquer technique to route the Deutsch's difficult channel in 19 tracks. However, Szymanski [8] showed that the channel routing problem with the introduction of doglegs is also NP-complete.

In recent years, some other greedy or hierarchical channel routers [9–11] were proposed to solve the channel routing problem. For example, a linear-time hierarchical router [9] gave a new greedy idea to solve a channel routing problem in a polynomial time. Two greedy channel routers [10, 11] further yielded an optimal result with 19 tracks for the Deutsch's difficult channel. In contrast to two-layer non-overlap Manhattan grid model, the channel routing problem in a multi-layer model [12–14], overlap model [14–16], diagonal model [17, 18] or gridless model [19, 20] has been extensively studied. Besides that, the crosstalk constraint between any pair of adjacent nets is further considered in the channel routing problem, and some minimum crosstalk channel routers [21, 22] are proposed.

In this paper, based on the hybrid methodology of top routing and bottom routing, we propose a linear-time approach for the channel routing problem. Basically, top (bottom) routing is a track-assignment-based routing approach in a channel, i.e. a channel is routed track by track from top (bottom) to bottom (top) by running top (bottom) routing. In the proposed routing approach, the routing process is divided into two phases: iterative-construction phase and merging-improvement phase. In the iterative-construction phase, the net interval of each routing net is split into horizontal segments, and these segments are further assigned track by track in a top-down or bottom-up manner. In the merging-improvement phase, the routing result is further improved by merging shorter segments in different tracks into longer segments for the reduction of the total wire length and the number of vias. Finally, the time complexity of the proposed approach is proven to be in $O(N_{\mathrm{col}})$ time, where N_{col} is the number of columns in a channel. In the experimental result, the proposed approach has tested many published channels and the routing results are in the optimal number of tracks. For example, the Deutsch's difficult channel is routed in 19 tracks with automatic introduction of doglegs. In addition to the optimality of the number of tracks, the proposed approach obtains fewer vias and shorter total wire length than all other Manhattan channel routers.

2. Problem description and definitions

Traditionally, the channel routing problem in a two-layer non-overlap Manhattan grid model can be summarized as follows.

A channel, C, is a rectangular routing region with two rows of fixed terminals, which are located at two opposite boundaries of this region and are represented by an ordered top terminal set, T, and an ordered bottom terminal set, B, respectively. The routing operations of all the nets are performed on virtual rectilinear grids that consist of vertical and horizontal grid lines and represent physical vertical and horizontal tracks. No wire segment is allowed outside the channel or along the diagonal direction. In general, there are two location functions, $t(i)$ and $b(i)$, existing in the channel boundaries, where $t(i)$ ($b(i)$) represents a net number marked on the ith column of the top (bottom) boundary. Basically, the routing process is performed on two independent routing layers. One layer, the *vertical layer*, carries all of the vertical wire segments and the other layer, the *horizontal layer*, carries all of the horizontal wire segments. These wire segments located on different layers are connected by vias. The fixed terminals located at the top and bottom channel boundaries are accessible on the vertical layer.

For the routing process, the same numbered terminals in top and bottom boundaries must be connected in a channel region. That means all of the routing nets, N_1, N_2, ..., N_n, whose endpoints are located on top and bottom boundaries must be routed in the channel region, C, where N_i represents a routing net i, $1 \leq i \leq n$. The objective of the channel routing problem is to route all the nets, $N_1, N_2, \ldots, N_n$, belonging to the channel, C, in a minimum channel width (minimum area without extending the channel length). Hence, the channel routing problem can be further described as:

Minimize the channel width $w = W(C, N_1, N_2, \ldots, N_n)$,

subject to two-layer non-overlap Manhattan grid model,

where $W(.)$ is a channel-width function based on a given channel and the connection of all the routing nets in this channel.

As shown in Fig. 1, the routing problem is easily expressed by a net-list representation, where arrows indicate whether these nets are to be connected to terminals on the upper or lower sides in this channel.

Definition 1. For a given net N_i, the net interval, L_i, is defined as a horizontal interval between leftmost and rightmost terminals in the net N_i, i.e. $L_i = [L(N_i), R(N_i)]$, where $L(N_i)$ ($R(N_i)$) is the horizontal coordinate of the leftmost (rightmost) terminal in the net N_i.

Definition 2. For a given channel C, the local density, D_i, at column i is defined as the number of net intervals which cross the ith column in the channel, i.e. $D_i = |\{N_j | L(N_j) \leq i \leq (N_j)\}|$, where $|S|$ represents the number of elements involved in a set S. Furthermore, the channel density, $D_{\max}$, is defined as the maximal local channel density in the channel, i.e. $D_{\max} = \max(D_i)$, where $1 \leq i \leq N_{\mathrm{col}}$, where N_{col} is the number of columns in the channel.

Definition 3. For a given channel C, a vertical constraint graph (VCG), G_{vc}, is defined as a directed graph $G_{\mathrm{vc}} = G(V, E)$, where a node n_i in V is defined as the net interval of the net N_i, and a directed edge $e_{i,j}$ connected from node n_i to node n_j represents that the net interval involved in n_i must be placed above the net interval involved in n_j.

In a two-layer non-overlap Manhattan grid model, there are only two routing layers for the channel routing problem and any two vertical (horizontal) wire segments do not overlap at any vertical (horizontal) column. Traditionally, any routing net in a channel is restricted to route its net interval on a horizontal track, i.e. the net interval is not split into sub-intervals and

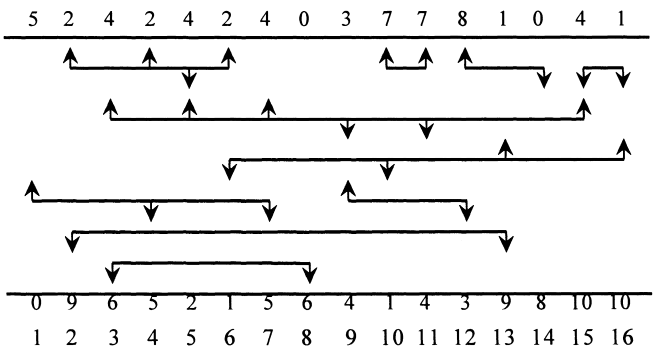


Fig. 1. Netlist representation for a channel routing specification.

assigned on different horizontal tracks. Hence, one net interval connected to the upper terminal at a given column must be placed above the other net interval connected to the lower terminal at the same column. Based on vertical constraints in a traditional manner, for a given channel, a vertical constraint graph (VCG) will be established for the channel routing problem. Furthermore, the channel density, $D_{\max}$, in the channel will be obtained according to the definition of the local density in the channel. In general, the channel density, $D_{\max}$, seems to be a lower bound of the number of tracks for the channel routing problem.

In order to minimize the number of tracks in a channel, this traditional restriction on the net interval will be relaxed by splitting any net interval into sub-intervals and introducing doglegs to connect sub-intervals on vertical tracks. Basically, the introduction of doglegs on vertical tracks is divided into the introduction of *restricted doglegs* and the introduction of *free doglegs*.
Definition 4: For a given channel, a dogleg is defined as one vertical wire segment whose endpoints are connected to the endpoints of two horizontal sub-intervals through vias. Furthermore, the introduction of a restricted dogleg is defined as a dogleg by which a net interval is split at the columns whose terminal is one terminal in this net. On the other hand, the introduction of a free dogleg is defined as a dogleg by which a net interval is split at any column in the channel.

Although most of the channels are successfully routed without the introduction of any dogleg, they will be routed in a minimum channel width if the restricted or free doglegs are introduced into these channels (refer to Fig. 1). Fig. 2 shows the routing result without the introduction of any dogleg, Fig. 3 shows the routing result with the introduction of restricted doglegs and Fig. 4 shows the routing result with the introduction of free doglegs.

Based on this traditional restriction on the net interval in a channel, the channel routing problem is solved by finding a routing ordering on horizontal tracks. Clearly, the routing ordering will be obtained by searching a vertical constraint graph G_{vc}, and these net intervals will be assigned in a topological order if G_{vc} is acyclic. However, the channel routing problem is not solved if G_{vc} is cyclic. In general, the cycles in a vertical constraint graph is divided into *pseudo cycles* and *real cycles*.
Lemma 1. *For a channel C without the introduction of any dogleg, if its vertical constraint graph*

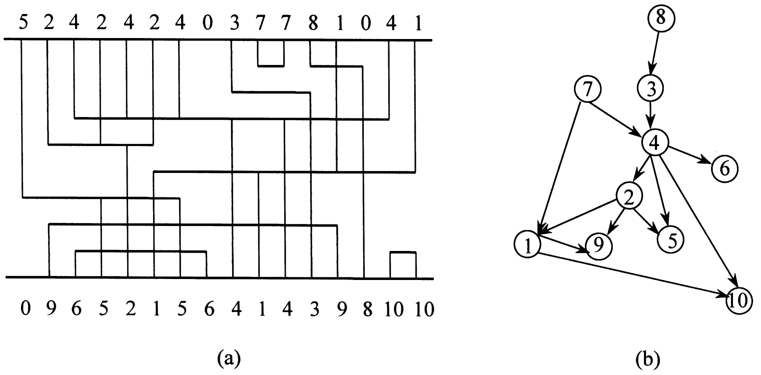


Fig. 2. (a) Routing result of Fig. 1 without the introduction of any dogleg. (b) Vertical constraint graph of Fig. 1.

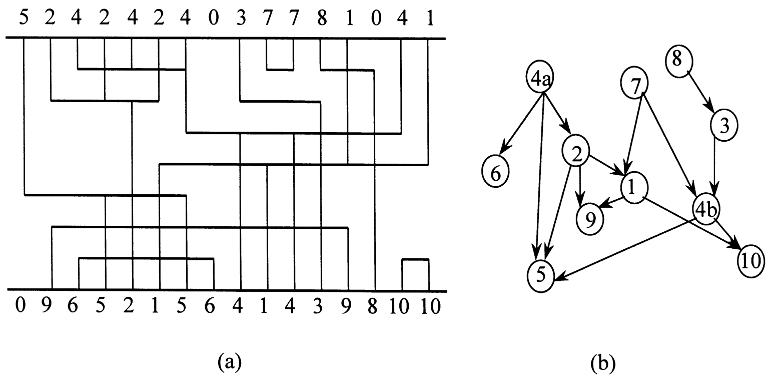


Fig. 3. (a) Routing result of Fig. 1 with the introduction of restricted dogleg. (b) Vertical constraint graph of (a).

is cyclic for the channel routing problem, the routing specification is not routable.

Definition 5. Given a vertical constraint graph G_{vc}, a pseudo cycle is defined as a cycle in G_{vc}, which can be broken by the introduction of a dogleg. On the other hand, a real cycle is defined as a cycle in G_{vc}, which cannot be broken by the introduction of a dogleg within the channel length.

As mentioned above, this traditional restriction on the net interval will be relaxed by splitting any net interval into sub-intervals and introducing doglegs to connect sub-intervals on vertical tracks. From the viewpoint of a vertical constraint graph G_{vc}, a vertex in G_{vc} will be split into two independent vertices by introducing a dogleg. In general, all of the pseudo cycles in G_{vc} will be broken by the introduction of restricted or free doglegs. Hence, the channel routing problem will be solved if G_{vc} is cyclic and all of the cycles in G_{vc} are pseudo cycles, i.e. the channel routing problem whose vertical constraint graph has only pseudo cycles will be solved by assigning feasible doglegs into vertical tracks. However, if a vertical constraint graph has real cycles, the channel routing problem will not be solvable by the introduction of restricted or free doglegs. For the channel routing problem in Fig. 5, there are two pseudo cycles, {3, 4} and {3, 4, 5}, in its vertical constraint graph. The net interval of nets, 3 and 4, will be split into two sub-intervals to break these two pseudo cycles, and the two vertices, 3

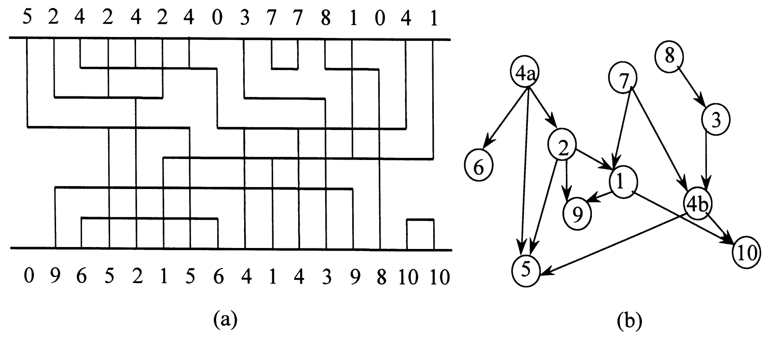


Fig. 4. (a) Routing result of Fig. 1 with the introduction of free dogleg. (b) Vertical constraint graph of (a).

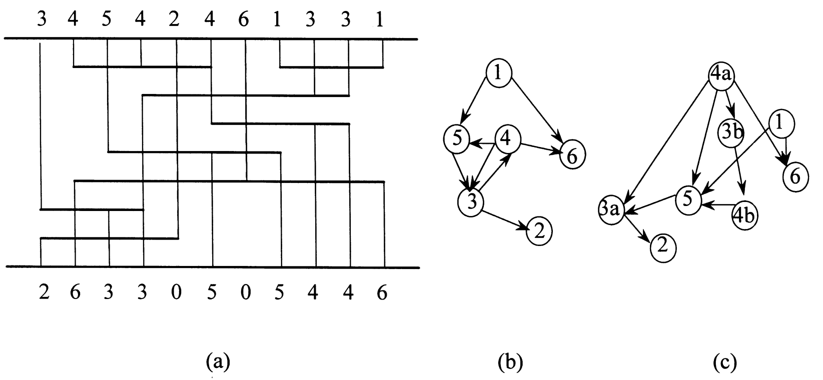


Fig. 5. Channel routing and the introduction of doglegs. (a) Routing result with the introduction of free doglegs. (b) Original vertical constraint graph with one pseudo cycle. (c) Vertical constraint graph of (a).

and 4, will be, respectively, replaced by two independent vertices in its vertical constraint graph. Finally, the routing result will be obtained by finding a routing ordering on horizontal tracks.

Because the proposed router is based on track assignment for top routing and bottom routing, some available definitions are described as follows.

Definition 6. Given a horizontal interval set I, an undirected graph $G = (V, E)$ is called an interval graph if each vertex in the vertex set, V, represents an interval in I, the vertex weight is assigned by the interval length, and each edge in the edge set, E, represents an intersection relation between any pair of intervals.

Definition 7. Given a channel C, the vacant-split weight, W_{vacant}, is defined as the sum of the number of net intervals split at any vacant column by a dogleg, such that

$$W_{\text{vacant}}(C) = \sum_{\substack{1 \le i \le N_{\text{col}}, \\ t(i)=0,\ \text{orb}(i)=0}} D_i,$$

where N_{col} is the number of columns in a channel.

Definition 8. For a given track, T_i, the efficiency, $R_{\text{efficiency}}[i]$, is defined as the ratio of the total wire length on T_i to the track length on T_i, i.e. $R_{\text{efficiency}}[i]$ = total_wire_length/track_length.

3. Track assignment for top routing and bottom routing

Basically, top routing (bottom routing) is a track-assignment-based routing approach from top (bottom) to bottom (top) in a channel. Hence, track assignment is a fundamental and important operation in top routing and bottom routing. In this section, one new approach is proposed to solve the track assignment problem in top routing and bottom routing. In general, the solution of the track assignment problem is two main steps: *candidate set construction* and *track assignment*. In our approach, two candidate interval sets on the topmost and bottommost tracks are, respectively, constructed in the *candidate set construction* step.

3.1. Candidate set construction

For a given channel, the construction of two candidate interval sets on the topmost and bottommost tracks depends on the structure of a vertical constraint graph, the locations of vacant terminals and the distribution of routing nets in this channel. Basically, the construction of a candidate interval set on the topmost (bottommost) track in top (bottom) routing is to construct a feasible interval set for the track assignment step on the topmost (bottommost) track.

In the *candidate set construction* step, for a given channel C, a vertical constraint graph is established according to the distribution of the net intervals of all the routing nets in this channel and the locations of vacant terminals is known by scanning vertical columns in this channel. Furthermore, the net interval of each routing net in this channel is split into sub-intervals according to the locations of vacant terminals on the top or bottom boundary and the vertical constraint graph is modified by replacing any split interval with sub-intervals. Finally, if one interval is not assigned by any vertical constraint, this interval will be selected into a candidate interval set, S_{tcis}, on the topmost track. On the other hand, if one interval does not assign any vertical constraint into other intervals, this interval will be selected into a candidate interval set, S_{bcis}, on the bottommost track. Hence, two candidate interval sets, S_{tcis} and S_{bcis}, on the topmost and bottommost tracks will be constructed in the *candidate set construction* step.

For the construction of candidate sets, the input data includes a given channel specification and the output data is two candidate interval sets, S_{tcis} and S_{bcis} on the topmost and bottommost tracks. The pseudo-code of this algorithm, *Construct_Candidate_Set*, for the construction of S_{tcis} and S_{bcis} on the topmost and bottommost tracks is described as follows:

```
Algorithm Construct_Candidate_Set
Input: a given channel specification
Begin

    Construct a vertical constraint graph;
    Find all the vacant terminals in the channel;
 For (all of the columns in the channel)

    Begin
       If (the terminal in this column is a vacant terminal on top (bottom) boundary)
          Split the net intervals crossing this column into adjacent sub-intervals;

    Else
       Split the net interval, whose net contains any terminal in this column, into two sub-
       intervals.

   Modify the vertical constraint graph by replacing any split interval with sub-intervals;
```

```
        End
    Select feasible intervals that are not assigned by any vertical constraint into $S_{\text{tcis}}$;
    Select feasible intervals that do not assign any vertical constraint into other intervals into $S_{\text{bcis}}$;
    End
```

Theorem 1. *Algorithm Construct_Candidate_Set constructs two candidate interval sets, S_{tcis} and S_{bcis} on the topmost and bottommost tracks. For a given channel, the number of elements in S_{tcis} (S_{bcis}) is at most $O(N_{col})$, and the time complexity of this algorithm is in $O(N_{col})$ time, where N_{col} is the number of columns in the channel.*

Proof. Given a channel C, there are n nets, N_1, N_2, ..., N_n, in this channel. Let T_i be the terminal number of N_i on top boundary, V_{top} be the number of vacant terminals on top boundary and T_{trivial} be the number of trivial nets (one trivial net is two-terminal net and these two terminals are located at the same column) in the channel. Clearly, $T_1+T_2+\ldots+T_n+V_{\text{top}}+T_{\text{trivial}}=N_{\text{col}}$, where N_{col} is the number of columns in this channel. By running the algorithm *Construct_Candidate_Set*, the net interval of each net is split into some sub-intervals according to the number of vacant terminals under its net interval and the number of terminals on top and bottom boundaries. For the construction of the candidate set S_{tics}, we define the function $F_{\text{top}}()$ to formulate the number of sub-intervals in N_i, such that

$$F_{\text{top}}(N_i)=T_i+V_i-1,\quad \text{if } T_i+V_i-1\geq 0,\qquad F_{\text{top}}(N_i)=0,\quad \text{if } T_i+V_i-1<0,$$

where V_i be the number of vacant terminals under the net interval of N_i in this channel. Hence, the number of intervals in the channel will be obtained as the sum of $F_{\text{top}}(N_1)$, $F_{\text{top}}(N_2)$, ..., $F_{\text{top}}(N_n)$, such that

$$\sum_{i=1}^{n}F_{\text{top}}(N_i)=\sum_{n_i+v_i-1\geq 0}(T_i+V_i-1)=\sum_{i=1}^{n}T_i+\sum_{i=1}^{n}V_i-m,$$

where m is the number of nets with $T_i+V_i-1>0$.

$$\sum_{i=1}^{n}F_{\text{top}}(N_i)=\sum_{i=1}^{n}V_i+N_{\text{col}}-V_{\text{top}}-T_{\text{trivial}}-m=\sum_{V_i=1}^{V_{\text{top}}}D_{c(vi)}+N_{\text{col}}-V_{\text{top}}-N_{\text{trivial}}-m,$$

where $c(v_i)$ is the column number of the ith vacant terminal in the channel.

$$\sum_{i=1}^{n}F_{\text{top}}(N_i)\leq V_{\text{top}}*D_{\max}+N_{\text{col}}-V_{\text{top}}-N_{\text{trivial}}-m=V_{\text{top}}*(D_{\max}-1)+N_{\text{col}}-N_{\text{trivial}}-m$$

In general, N_{trivial} is much fewer than N_{col} and m is at most N_{col}. Hence, (N_{col}–N_{trivial}–m) is $O(N_{\text{col}})$ in the worst case. On the other hand, $D_{\max}$ will be $O(1)$ as V_{top} is $O(N_{\text{col}})$ and V_{top} will be $O(1)$ as $D_{\max}$ is $O(N_{\text{col}})$. Clearly, $V_{\text{top}}*(D_{\max}-1)$ is $O(N_{\text{col}})$ for a practical channel. Hence, the number of net intervals in the channel is at most $O(N_{\text{col}})$. For the construction of a candidate set S_{tcis}, the number of elements in S_{tcis} is at most $O(N_{\text{col}})$ in a channel. Similarly, the number of elements in S_{bcis} is at most $O(N_{\text{col}})$ in a channel.

In the algorithm *Construct_Candidate_Set*, the time complexity of constructing a vertical constraint graph and finding all of the vacant terminals in a channel is in $O(N_{\text{col}})$ time.

Furthermore, the iteration in a for-loop statement is N_{col} and the time complexity in the loop-body is in $O(1)$ time. Hence, the time complexity of the for-loop statement is in $O(N_{col})$ time. Finally, the time complexity of constructing two candidate sets S_{tcis} and S_{bcis} is in $O(N_{col})$ time. Therefore, the time complexity of the algorithm *Construct_Candidate_Set* is in $O(N_{col})$ time. □

3.2. Track assignment for a candidate set

After constructing two candidate interval sets on the topmost and bottommost tracks, the track assignment process on the topmost (bottommost) track is to get a subset of a candidate interval set in which there exists no intersection relation between any pair of intervals. In the track assignment process, each interval in a candidate interval set is assigned by a weight value and an interval graph is established according to the intersection relation between any pair of intervals. In Fig. 6, given an interval set, an interval graph is constructed and illustrated. Finally, the problem of finding a subset of the candidate interval set on the topmost or bottommost track corresponds to the problem of finding a maximum-weight independent set in the interval graph.

In the proposed approach, if two intervals belong to the same net, there will be no intersection relation between the two intervals for the construction of an interval graph. For each interval in S_{tcis} (S_{bcis}), if the interval length is used as the weight, the track assignment process will be obtained by running a maximum-weight independent set algorithm. To our knowledge, the problem of finding a maximum-weight independent set in an interval graph can be efficiently solved by running the algorithm MWIS [23].

Theorem 2. *For the interval sets, S_{tcis} and S_{bcis}, the algorithm MWIS finds a maximum-weight independent set in an interval graph. The time complexity of the algorithm is in $O(N_{col})$ time, where N_{col} is the number of columns in a channel.*

Proof. By the proof of the algorithm MWIS [23], the time complexity of the algorithm MWIS is in $O(n)$ time, where n is the number of intervals in an interval graph. By Theorem 1, the number of intervals in S_{tcis} and S_{btis} is $O(N_{col})$. Therefore, the time complexity of the algorithm MWIS will be in $O(N_{col})$ time. □

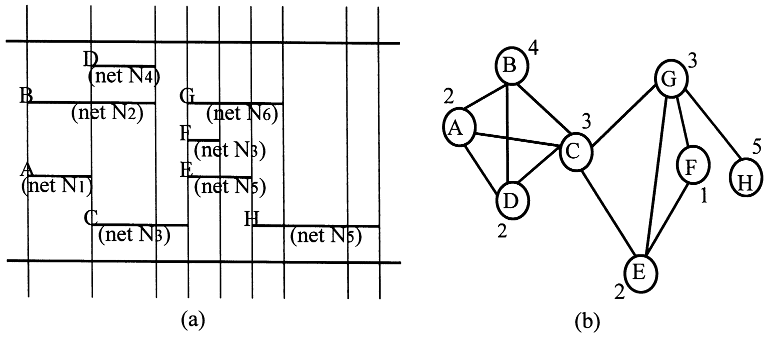


Fig. 6. (a) Zone representation of horizontal intervals in a channel. (b) Interval graph of (a).

Now, two maximum-weight independent sets in S_{tcis} and S_{bcis} are ready to assign the intervals on the topmost and bottommost tracks, respectively. In the proposed approach, only one independent set is used to assign the intervals on the topmost or bottommost track for the track assignment process. Hence, three weight values, Weight1, Weight2 and Weight3, will be further defined to decide which independent set is used assign the intervals on a track as follows:

Weight1: reduction of the channel density in a channel, $D_{\mathrm{reduction}}$.
Weight2: increment of the vacant-split value in a channel. $W_{\mathrm{increment}}$; $W_{\mathrm{increment}} = W_{\mathrm{vacant}}(C_{\mathrm{modified}}) - W_{\mathrm{vacant}}(C_{\mathrm{original}})$, where C_{original} (C_{modified}) is a channel before (after) the track assignment process on the topmost or bottommost track.
Weight3: efficiency of a track for the track assignment process, $R_{\mathrm{efficiency}}$.

According to the definitions of the weights, Weight1, Weight2 and Weight3, two independent sets are assigned by the weight value (Weight1, Weight2, Weight3), respectively. Furthermore, the problem of deciding which independent set is used assign the intervals on a track is determined by comparing the weight value (Weight1, Weight2, Weight3) in a lexicographical order. Finally, all of the intervals in this independent set will be assigned on the topmost or bottommost track in the channel.

For the track assignment process on the topmost or bottommost track, the input data includes a given channel specification and the output data is the track assignment on the topmost or bottommost track. The pseudo-code of this algorithm, *Track_Assignment*, for the track assignment process is described as follows:

```
Algorithm Track_Assignment
Input: a given channel specification;
Begin

 Step 1: call algorithm Construct_Candidate_Set to build $S_{tcis}$ and $S_{bcis}$;
 Step 2: construct two interval graphs according to $S_{tcis}$ and $S_{bcis}$, respectively;
 Step 3: assign the weight value on each interval in $S_{tcis}$ and $S_{bcis}$;
 Step 4: call algorithm MWIS for $S_{tcis}$ and $S_{bcis}$ to obtain two maximum weight
         independent sets;
 Step 5: compare two maximum weight independent sets according to the weight value,
         (Weight1, Weight2, Weight3), in a lexicographical order;
 Step 6: return the maximum weight independent set with greater weight and assign on its
         related track;

End
```

Theorem 3. *Given a channel, the algorithm Track_Assignment can complete the track assignment on the topmost or bottommost track in the channel. The time complexity of the algorithm Track_Assignment is in $O(N_{col})$ time, where N_{col} is the number of columns in a channel.*

Proof. By Theorem 1, the time complexity of the algorithm *Construct_Candidate_Set* is in O(N_{col}) time. Again, two interval graphs for S_{tcis} and S_{btis} are constructed by scanning all of the columns in this channel, hence the time complexity of this step is in $O(N_{\mathrm{col}})$ time. By Theorem 1, the number of intervals in S_{tcis} and S_{bcis} is $O(N_{\mathrm{col}})$. The time complexity of the weight assignment for S_{tcis} and S_{bcis} is in $O(N_{\mathrm{col}})$ time. Furthermore, by Theorem 2, the time complexity of the algorithm MWIS is in $O(N_{\mathrm{col}})$ time. The time complexity of assigning the weight (Weight1, Weight2, Weight3) for two independent sets and comparing the two independent sets is also in $O(N_{\mathrm{col}})$ time. Finally, the time complexity of the track assignment for an independent set is in $O(N_{\mathrm{col}})$ time. Therefore, the time complexity of the algorithm *Track_Assignment* is in $O(N_{\mathrm{col}})$ time. □

4. Designing a track-assignment-based channel router

Basically, the proposed channel router based on the hybrid methodology of top routing and bottom routing is divided into two phases: *iterative-construction* phase and *merging-improvement* phase. In the *iterative-construction* phase, the track assignment process will assign feasible intervals on the topmost or bottommost track one by one. Until all of the net intervals in the channel are fully assigned on the tracks, a routing result based on the hybrid methodology of top routing and bottom routing will be obtained. Furthermore, the total wire length or the number of vias in the routing result is improved by merging adjacent intervals into one longer interval in the *merging-improvement* phase.

4.1. Iterative-construction phase

In the *iterative-construction* phase, the work is to obtain a routing result based on the hybrid methodology of top routing and bottom routing. Basically, the main operation in the iterative-construction phase is one track assignment process on the topmost or bottommost track in the channel. As mentioned above, given a channel, the track assignment process on the topmost or bottommost track is completed by the algorithm *Track_Assignment*. As the track assignment process on the topmost or bottommost track is completed, the corresponding vertical trunks will be connected to these intervals from the terminals on the top or bottom boundary and a new channel will be obtained by defining the terminals on the top or bottom boundary. During the track assignment process, the restricted or free doglegs will be introduced to break all the pseudo cycles in the channel routing problem by connecting these vertical trunks. Until all of the net intervals are fully assigned on the tracks in the channel track by track, a routing result will be obtained in the iterative-construction phase. Clearly, the routing result is the integration of the result of top routing and the result of bottom routing.

Referring to Fig. 1, a routing result will be obtained in the iterative-construction phase. The result of top routing and the result of bottom routing are shown in Fig. 7(a) and (b), respectively. An initial routing result for the channel routing problem is illustrated in Fig. 7(c).

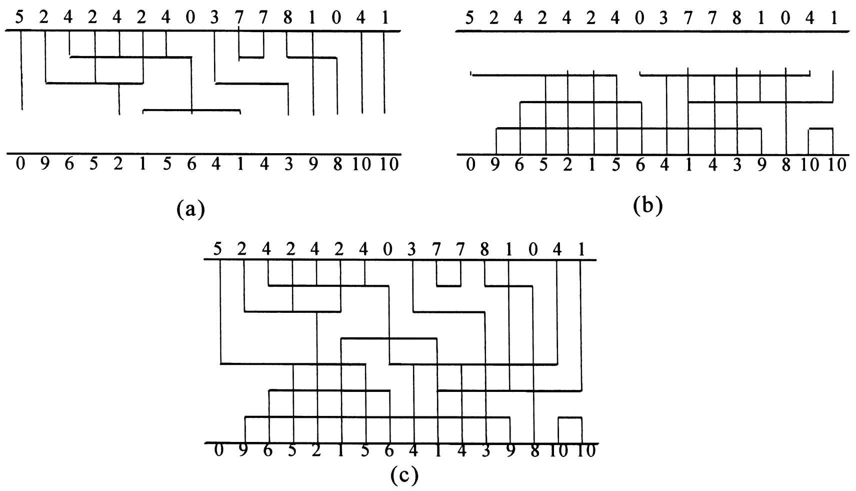


Fig. 7. Results of top routing and bottom routing. (a) Routing result in top routing. (b) Routing result in bottom routing. (c) A routing result after the iterative-construction phase.

4.2. Merging-improvement phase

In the *merging-improvement* phase, given a routing result, the work is to reduce the total wire length or the number of vias by merging two adjacent intervals into one longer interval. Basically, the initial routing result is obtained by using the hybrid methodology of top routing and bottom routing. Because top routing and bottom routing are track-assignment-based approaches, the net interval of each routing net may be split into many sub-intervals and these sub-intervals may be assigned on different tracks in the channel. In general, unnecessary doglegs will yield unnecessary vias and increase the total wire length in a channel. Hence, these unnecessary doglegs will be deleted by merging adjacent intervals into one longer interval.

Basically, the merging operation is divided into *full-merging* operation and *partial-merging* operation. In one *full-merging* operation, an unnecessary dogleg is eliminated by fully merging two adjacent intervals into one longer interval. Hence, the total wire length will be reduced and two vias will be deleted in this *full-merging* operation. On the other hand, the total wire length or the number of vias will be reduced by partially merging two adjacent intervals into one longer interval in one *partial-merging* operation. Hence, if one *full-merging* operation is not run between the pair of adjacent intervals, one improvement on the total wire length or the number of vias will be obtained by using one *partial-merging* operation. One full merging operation and one partial merging operation in a routing net are shown in Fig. 8.

4.3. Design of a track-assignment-based channel router

As mentioned above, if there exists any real cycle in a vertical constraint graph, no vertex in the graph will be split to break the real cycle, and no restricted or free dogleg will be introduced into the channel routing problem. Hence, the channel routing problem is not solved within the channel length in two-layer non-overlap Manhattan grid model. In the following, we formulate an unroutable channel routing problem. By adding a pair of external vacant

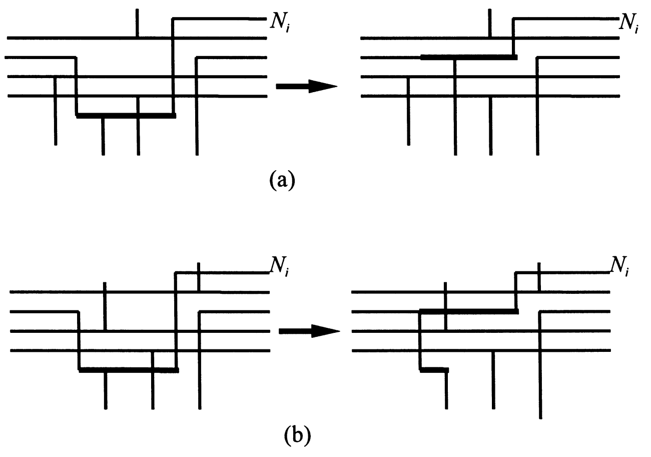


Fig. 8. Merging operation in the merging-improvement phase. (a) One full-merging operation. (b) One partial-merging operation.

terminals in the channel, the channel introduces external doglegs to break the real cycles in the vertical constraint graph.

Lemma 2. *For a given channel, if the channel routing problem satisfies the following two conditions,*

1. All of the routing nets belong to one-to-one net,
2. There is no internal vacant terminal located on the top or bottom boundary in the channel,

this channel will not be routable within the channel length.

For the proposed channel router based on the hybrid methodology of top routing and bottom routing, the input data includes a given channel specification and the output data is the routing result in the channel. Before solving the channel routing problem, An unroutable condition will be checked whether this problem is unroutable. If this problem is unroutable, a pair of external vacant terminals outside the channel will be added to break the real cycle in a vertical constraint graph. In fact, the external vacant column is used to yield a necessary detoured wire path and introduce a dogleg to break the real cycle. The pseudo-code of this algorithm, *channel_Router*, for the track-assignment-based channel router is described as follows:

```
Algorithm Channel_Router
Input: a given channel specification;
Begin
Step 1: if the unroutable condition for the channel routing problem is satisfied, then
        1. Extend the channel length by adding an external vacant terminal column;
        2. Extend the longer net interval to an external vacant terminal column;
        3. Modify, the net intervals of routing nets and the location of vacant terminals;
```

```
Step 2: consider whether all of the net intervals are assigned;
        If it is, then Goto Step 6;
Step 3: modify the vertical constraint graph, $G_{vc}$, the net intervals of routing nets and the
        location of vacant terminals;
Step 4: call algorithm Track_Assignment for the track assignment on the topmost track or
        the bottommost track;
Step 5: Extend and assign vertical trunks to the assigned intervals;
        Modify the original channel by defining the terminals on the top boundary or the
        bottom boundary;
Goto Step 2;
Step 6: for (any pair of adjacent intervals) Merge the pair of adjacent intervals by one
        full-merging operation or one partial-merging operation;
  End
```

Theorem 4. *For a given channel, the algorithm Channel_Router based on the hybrid methodology of top routing or bottom routing will solve the channel routing problem. The time complexity of the algorithm Channel_Router is in $O(N_{col})$ time, where Ncol is the number of columns in the channel.*

Proof. In the algorithm, *Channel_Router*, Step 1 can be completed by scanning all of the columns in a channel. Hence, the time complexity of the step is in $O(N_{\text{col}})$ time. The iterative number in this loop-statement from Step 2 to Step 5 is $O(D_{\max})$, so the time complexity of the track assignment process in the iterative-construction phase is in $O(D_{\max}*N_{\text{col}})$ time. Finally, the time complexity of Step 6 is in $O(N_{\text{col}})$ time. Therefore, the time complexity of the algorithm *Channel_Router* is in $O(D_{\max}*N_{\text{col}})$ time. In general, $D_{\max}$ is much less than N_{col} in a channel, i.e. $O(D_{\max}) \approx O(1)$. Therefore, the time complexity of the algorithm *Channel_Router* is in $O(N_{\text{col}})$ time. □

5. Experimental results

The proposed track-assignment-based channel router *Channel_Router* has been implemented using standard C language and run on a SUN SPARC workstation under the Berkeley 4.2 UNIX operating system. By applying the proposed track-assignment-based channel router to many known channels, ex1, ex2, ex3b, ex3c, ex4b, ex5 and diff, it turns out that the proposed channel router generates optimal routing results in the number of tracks for these known channels. In Table 1, the routing results of the channel routers, Left-Edge [1], Efficient [5] and Robust [11] for the known channels are listed and compared with the routing result of the proposed channel router. For example, for the Burstein's difficult channel, if no vacant column is added into the channel, the proposed channel router will obtain a routing result in 8 tracks. To our knowledge, without extending the channel length, such a routing result in two-layer non-overlap Manhattan grid model is optimal. Furthermore, if the Burstein's difficult channel is added one vacant column into the channel, the proposed channel router will obtain a routing result in 6 tracks. We believe that the routing result is optimal in two-layer Manhattan grid model without extending the length of the channel. For the Burstein's difficult channel, the

Table 1
Routing results for channel benchmarks

Example (#net)	#Columns	Channel density	Left-edge [1]	Efficient [5]	Robust [11]	Our channel router (time: s)
ex1(21)	35	12	14	12	12	12 (0.32)
ex2(30)	62	15	18	15	15	15 (0.53)
ex3b(47)	61	17	20	17	17	17 (0.61)
ex3c(54)	79	18	19	18	18	18 (0.86)
ex4b(57)	119	17	23	17	17	17 (1.31)
ex5(62)	119	20	22	20	20	20 (1.47)
diff(72)	174	19	39	20	19	19 (2.13)

routing results with and with adding one vacant column into the channel are as shown in Fig. 9.

In general, the Deutsch's difficult example is used as the standard benchmark in estimating the performance of a channel router. Now, the proposed channel router is used to route the Deutsch's difficult example in two-layer non-overlap Manhattan grid model. It obtains a routing result in 19 tracks as illustrated in Fig. 10. In addition to maintaining the optimality of routing tracks in the channel, the routing result was qualified through the total wire length and the number of vias in the channel. For the Deutsch's difficult example, the proposed channel router generates 318 vias and 4959 units of routing wires in two-layer non-overlap Manhattan grid model. For the experimental result in the Deutsch's difficult channel, the proposed channel router yields shorter total wire length and fewer vias than some other Manhattan channel routers in Table 2. Clearly, the proposed channel router indeed has an improvement on both the total wire length and the number of vias for the Deutsch's difficult example.

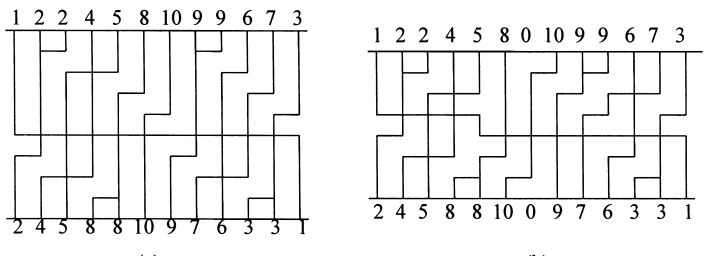


Fig. 9. Burstein's difficult channel. (a) Routing result without one empty column. (b) Routing result with one empty column.

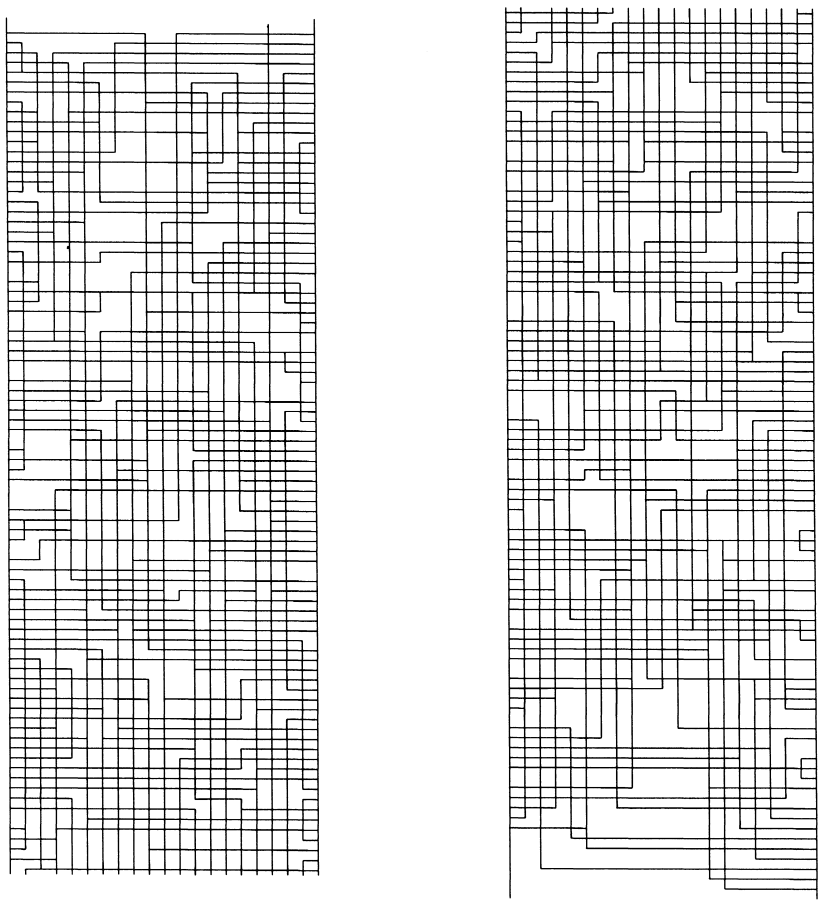


Fig. 10. Routing result for Deutsch's difficult example.

Table 2
Comparison of routing result in Deutsch's difficult example

Router	Horizontal wire on both layers	Vertical wire on both layers	Tracks	Vias	Net length
YARC2 [15]	yes	no	19	287	5020
MIGHTY [16]	yes	yes	19	301	4812
Burstein's [7]	no	no	19	336	5023
Ho's [10]	no	no	19	333	5004
Robust router [11]	no	no	19	319	4961
Our router	no	no	19	318	4959

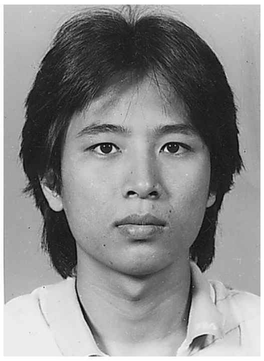
6. Conclusion

In this paper, based on the hybrid methodology of top routing and bottom routing, we propose an $O(N_{\text{col}})$ approach for the channel routing problem, where N_{col} is the number of columns in a channel. Basically, top (bottom) routing is a track-assignment-based routing approach in a channel, that is, a channel is routed track by track from top (bottom) to bottom (top) by running top (bottom) routing. In the proposed routing approach, the routing process is divided into two phases: iterative-construction phase and merging-improvement phase. In the iterative-construction phase, the net interval of each routing net is split into horizontal segments and these segments are further assigned track by track in a top-down or bottom-up manner. In the merging-improvement phase, the routing result is further improved by merging shorter segments in different tracks into longer segments for the reduction of the total wire length and the number of vias. Finally, the proposed approach has tested many published channels and the routing results are in the optimal number of tracks. For example, the Deutsch's difficult channel is routed in 19 tracks with automatic introduction of doglegs. In addition to the optimality of the number of tracks, the proposed approach obtains fewer vias and shorter total wire length than all other Manhattan channel routers.

References

[1] Hashimoto A, Stevens J. Wire routing by optimizing channel assignment within large apertures. Proc. 8th Design Automation Workshop. 1971. p. 155–69.
[2] Persky G, Deustch DN, Schweikert DG. LTX: a minicomputer-based system for automatic LSI layout. J Design Automation Fault-Tolerant Comput 1977:1217–55.
[3] LaPaugh AS. Algorithms for integrated circuit layout: an analytic approach. Ph. D. Dissertation, Laboratory for Computer Science, MIT, 1980.
[4] Deutsch DN. A dogleg channel router. Proc. of the 13th Design Automation Conference. 1976. p. 425–33.
[5] Yoshimura T, Kuh ES. Efficient algorithms for channel routing. IEEE Trans Comput Aided Design 1982;1:25–35.
[6] Rivest RL, Fiduccia CM. A greedy channel router. Proc. 19th Design Automation Conference. 1982. p. 418–24.
[7] Burstein M, Pelavin R. Hierarchical wire routing. IEEE Trans Comput Aided Design 1983;2:223–34.
[8] Szymanski TG. Dogleg channel routing is NP-complete. IEEE Trans Comput Aided Design 1985;4:31–40.
[9] Hachtel GD, Morrison CR. Linear complexity algorithms for hierarchical routing. IEEE Trans Comput Aided Design 1989;8:64–80.
[10] Ho T-T, Sitharama Iyengar S, Zheng S-Q. A general greedy channel routing algorithm. IEEE Trans Comput Aided Design 1991;10:204–11.
[11] Yoeli U. A robust channel router. IEEE Trans Comput Aided Design 1991;10:212–9.
[12] Preparata FP, Lipski W, Jr. Optimal three layer channel routing. IEEE Trans Comput 1984;33:5.
[13] Pitchumani V, Zhang Q. A mixed HVH–VHV algorithm for three-layer channel routing. IEEE Trans Comput Aided Design 1987;6:497–502.
[14] Brady ML, Brown DJ. Optimal multilayer channel routing with overlap. Algorithmica 1991;6:83–101.
[15] Reed J, Sangiovanni-Vincentelli A, Santomauro M. A new symbolic channel router: YACR2. IEEE Trans Comput Aided Design 1985;4:208–19.
[16] Shin H, Sangiovanni-Vincentelli A. A detailed router based on incremental routing modifications: mighty. IEEE Trans Comput Aided Design 1987;6:942–55.
[17] Chaudhary K, Robinson P. Channel routing by sorting. IEEE Trans Comput Aided Design 1991;10:754–60.

[18] Wang DC. Novel routing schemes for IC layout. Part I: two layer channel routing. Proc. of the 28th Design Automation Conference. 1991. p. 49–53.
[19] Chen HH, Kuh ES. Glitter: a gridless variable-width channel router. IEEE Trans Comput Aided Design 1986;5:459–65.
[20] Groeneveld P. A multiple layer contour-based gridless channel router. IEEE Trans Comput Aided Design 1990;9:1278–88.
[21] Gao T, Liu CL. Minimum crosstalk channel routing. IEEE Trans Comput Aided Design 1996;15:465–74.
[22] Thakur S, Chao KY, Wong DF. An optimal layer assignment algorithm for minimizing crosstalk for three layer VHV channel routing. International Symposium on Circuits and Systems. 1995. p. 207–10.
[23] Chang JS, Tang CY, Chang RS. An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. Information Process Lett 1992;43:229–35.



Jin-Tai Yan received the B.Sc., M.Sc. and Ph.D. degrees in computer and information science from National Chiao-Tung University, Hsinchu, Taiwan, ROC in 1988, 1990 and 1995, respectively. From 1995 to 1997 he served in the Chinese Navy, Kaohsiung, Taiwan, as an information officer. Since 1997 he has been with the National Chaio-Tung University, Hsinchu, Taiwan and is currently a post-doctor researcher in Computer Systems Research Center. His current research interests are high-level synthesis, logic synthesis and physical design of VLSI circuits, parallel/distributed computing and network performance analysis.