Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

AN EFFICIENT MULTILAYER MCM ROUTER BASED ON FOUR-VIA ROUTING

K.-Y. Khoo
J. Cong

February 1993
CSD-930001

An Efficient Multilayer MCM Router
Based on Four-Via Routing

Kei-Yong Khoo and Jason Cong

Department of Computer Science
University of California at Los Angeles
Los Angeles, CA 90024

Abstract

In this paper, we present an efficient multilayer general area router, named V4R, for MCM and dense PCB designs. One unique feature of the V4R router is that it uses no more than four vias to route every net and yet produces high quality routing solutions. Another unique feature of the V4R router is it combines global routing and detailed routing in one step and produces high quality detailed routing solutions directly from the given netlist and module placement. Several combinatorial optimization techniques, including efficient algorithms for computing a maximum weighted k-cofamily in a partially ordered set and a maximum weighted non-crossing matching in a bipartite graph, are used to solve the combined problem efficiently. As a result, the V4R router is independent of net ordering, runs much faster, and uses far less memory compared to other multilayer general area routers. We tested our router on several examples, including three industrial MCM designs from MCC. Compared with the 3D maze router, on average the V4R router uses 44% fewer vias, 2% less wirelength, and runs 26 times faster. Compared with the SLICE router, on average the V4R router uses 9% fewer vias, 4% less wirelength, and runs 3.5 times faster. The V4R also uses fewer routing layers compared to the 3D maze router and the SLICE router.

1. Introduction

As VLSI fabrication technology advances, interconnection and packaging technology has become a bottleneck in system performance [PrPC89, Ba90, He90]. The multichip module (MCM) technology has been developed recently to increase the packing density and eliminate a level of interconnection by assembling and connecting bare chips on a common substrate. The substrate consists of multiple routing layers used for chip-to-chip interconnections. Without the individual packaging for the chips, the bare chips can be placed much closer on the MCM substrate, which leads to a significant increase in packing density and reduction in interconnection lengths.

Because of the high packing density in MCM designs, the MCM routing problem is more difficult than the conventional IC or PCB routing problems. First, MCMs may have far more interconnection layers than ICs. For example, the multi-chip module developed for the IBM 3081 mainframe has 33 layers of molybdenum conductors (including 1 bonding layer, 5 distribution layers, 16 interconnection layers, 8 voltage reference layers, and 3 power distribution layers[BlBa82, Bl83]). Fujitsu's latest supercomputer, the VP-2000, uses a ceramic substrate with over 50 interconnection layers[HaYY90]. Moreover, unlike routing in ICs where the routing region can be naturally decomposed into channels and switchboxes, there is no natural routing hierarchy in MCM routing. The MCM routing problem is an immense three-dimensional general area routing problem where routing can be carried out almost everywhere in the entire multilayer substrate. Finally, the line spacing is much smaller and the routing result is much denser in MCM routing as compared to those of conventional PCB routing. Thus, traditional PCB routing tools are often inadequate in dealing with MCM designs[1].

Few methods are available for multilayer MCM routing. A commonly used method for multilayer MCM designs is the three-dimensional (3D) maze routing [HaYY90, Mi91]. Although this method is conceptually simple to implement, it suffers from several problems. First, the quality of the maze routing solution is very sensitive to the ordering of the nets being routed, yet there is no effective algorithm for determining a good net ordering in general. Moreover, since each net is routed independently, global optimization is difficult and the final routing solution often uses a large number of vias. Finally, 3D maze routing requires long computational time and large memory space since it needs to store the entire routing grid and search in it. For example, one industrial MCM example that we obtained from MCC has a 45-micron routing pitch and a routing area of $152.4 \times 152.4\ mm^2$, which results in a routing grid of 3386×3386 (= 11,464,996 grid points) for a single layer and our routing solution consists of 4 layers. It is certainly not a trivial task for a 3D maze router to store such a routing grid and search in it efficiently.

Another method for multilayer MCM routing is to divide the routing layers into several $x-y$ layer pairs. Nets are first assigned to $x-y$ layer pairs and then two-layer routing is carried out for each $x-y$ layer pair (the x-layer runs horizontal wires and the y-layer runs vertical wires) [HoSV90]. Although this approach is efficient in general, it faces a few problems. First we have to pre-determine the number of the routing layers before we can carry out layer assignment, but there is no accurate estimation for the number of routing layers required. Moreover, detailed routing information, such as constraints on via and segment locations, are not considered during the layer assignment stage, which may lead to poor detailed routing results.

Recently, a multilayer MCM router named SLICE was developed by Khoo and Cong [KhCo92]. It computes a routing solution on a layer-by-layer basis and carries out planar routing in each layer. On average it uses 29% fewer vias and runs four times faster than the 3D maze router. However, since planar routing can complete only a limited number of nets, a two-layer maze router was used at each layer to complete as many remaining nets as possible. The use of maze router again slows down the computation and introduces extra vias.

[1] Besides the problem of efficient utilization of routing resource, there are also several performance issues involved in MCM routing. For example, for high-performance designs, the wires need to be modeled as lossy transmission lines, where signal reflection and cross-talk need to be taken into consideration.

Several efficient routers have been proposed for silicon-on-silicon based MCM technology [PrPC89, DaKJ90, DaDS91, DaKS91]. Since the number of routing layers is usually small (2 layers for signal routing in most cases) in this technology, many techniques for IC routing, such as hierarchical routing and rubber-band routing, can be applied to yield good solutions. However, it is not clear how to generalize these techniques to multilayer general area routing.

In this paper, we present an efficient multilayer general area router, named V4R, for MCM and dense PCB designs. One unique feature of the V4R router is that it uses no more than four vias to route every two-terminal net[2] and yet produces very satisfactory routing solutions. Marek-Sadowska [Ma84] showed a theoretical results that each two-terminal net can be routed using at most one via in a two-layer topological routing solution (this result was later generalized to multilayer topological routing by Rim, Kashiwabara, and Nakajima [RiKN89] and by Cong and Liu [CoLi90]). Although her result is interesting in theory, the resulting topological solution using one-via routing usually uses long wires and introduces congestion when mapped to a physical routing solution. Therefore, the method in[Ma84] is usually not applied directly in practice. To our knowledge, the V4R router is the first practical multilayer general area router that guarantees to use no more than a fixed number of vias for every net yet produces high quality physical routing solutions. Bounding the number of vias per net is not only helpful for via minimization but also very important for precise delay estimation at the higher level of MCM designs. For high-performance MCMs, vias not only increase the manufacture cost but also degrade the system performance since they form impedance discontinuities and cause reflections when the interconnections have to be modeled as transmission lines[Ba90].

Another unique feature of the V4R router is that it combines global routing and detailed routing in one step and produces high quality *detailed* routing solutions directly from the given netlist and module placement. Several combinatorial optimization techniques, including computing a maximum weighted k-cofamily in a partially ordered set and a maximum weighted non-crossing matching in a bipartite graph, help us to solve the combined problem efficiently. As a result, the V4R router is independent of net ordering, runs much faster, and uses far less memory. We tested our router on several examples, including three industrial MCM designs from MCC. Compared with the 3D maze router, on average the V4R router uses 44% fewer vias, 2% less wirelength, and runs 26 times faster. Compared with the SLICE router, on average the V4R router uses 9% fewer vias, 2% less wirelength, and runs 3.5 times faster. The V4R router also uses fewer routing layers as compared to the 3D maze router and the SLICE router.

The remainder of this paper is organized as follows. Section 2 formulates the multilayer MCM routing problem. In Section 3, we first give an overview of our algorithm and then describe each step of the algorithm in detail. Experimental results and comparative study are presented in Section 4.

[2] The majority of the nets in MCM designs are two-terminal nets. For example, in the MCM example of 37 VHSIC gate-arrays that we obtained from MCC, 94% of the nets are two terminals nets. A k-terminal net can be decomposed into $k-1$ two-terminal nets so that it can be routed using at most $4(k-1)$ vias by the V4R router.

2. Problem Formulation

The MCM routing problem consists of a set of modules, a set of nets, and a multilayer routing substrate. Modules (dies) are mounted on the top of the substrate by wire bonding, tape-automated bonding (TAB), or flip-chip bonding with solder bump connections. The substrate consists of multiple signal routing layers, with (possible) obstacles in some routing layers, such as power/ground connections and thermal conducting vias. The I/O terminals (pads) of the modules are connected to the substrate either directly or through routing to the external pads that surround the individual modules for engineering changes [Ba90]. The pads are brought to the first signal routing layer either directly through distribution vias or through one or more redistribution layers. The redistribution layers are required when the pad spacing does not match the line spacing on the signal routing layers. The pin redistribution problem is not studied in this paper, and the reader may refer to [ChSa91] for the solutions to the pin redistribution problem. The goal of our MCM router is to complete the connections for the I/O terminals in each net using the signal routing layers in the substrate.

The signal routing layers in the substrate are numbered from top to bottom. We assume that there is a Manhattan routing grid superimposed on each routing layer where the spacing between grid lines is determined by the routing pitch for the given technology. Two wires in adjacent signal routing layers can be connected by a via. Vias may be stacked on top of each other to connect wires in non-adjacent layers. Stacked vias can be formed in several ways, e.g., by filling the etched via with nickel in the AT&T AVP process or by plating copper posts as in the MCC process [Sh91]. Figure 1 shows a cross section of a sample four layer MCM routing substrate.

The output of the routing problem is a set of routing segments and vias that connect all the nets. The quality of the routing can be measured by the total wirelength, the number of vias, the number of wire bends (jogs) and the number of layers required to complete the routing. Long wire paths increase propagation time and should be avoided. Vias and wire bends degrades the signal's fidelity by introducing impedance discontinuity in signal paths thus should also be minimized. Each additional routing layer increases the manufacturing cost and thus the number of layers should also be minimized.

Now we introduce a few terminologies. In each routing layer, a horizontal grid line is referred to as a *horizontal track* and a vertical grid line is referred to as a *vertical track*. The terminals on

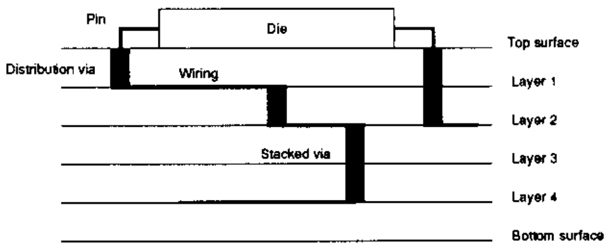

Figure 1: Cross section of a multilayer (4 layer) routing region

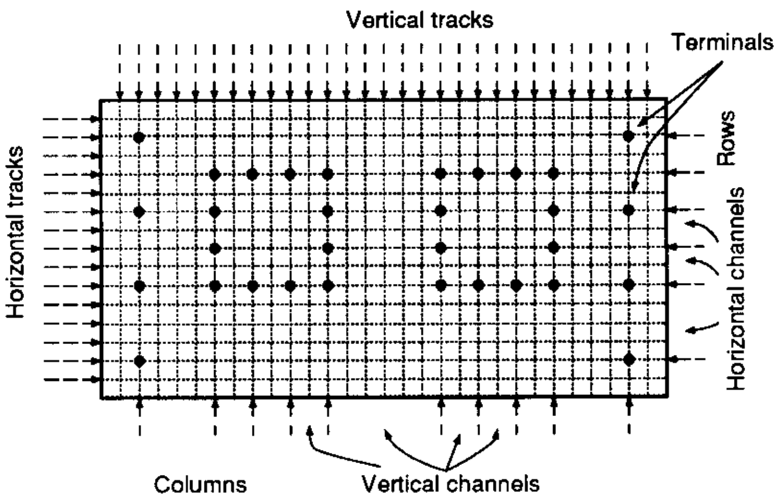

Fig. 2 Illustration of the definitions.

the same horizontal track form a *row*, and the terminals on the same vertical track form a *column*. For a terminal p, let $x(p)$ and $y(p)$ denote the x and y coordinates (in terms of grid point coordinates) of p respectively, and let $row(p)$ and $col(p)$ denote the row number and the column number of p respectively. Two adjacent rows form a *horizontal channel*, and two adjacent columns form a *vertical channel*. (See Fig. 2.) The channel formed between between the i-th and $(i+1)$-th rows (columns) is named the i-th horizontal (vertical) channel.[3] The *capacity* of a horizontal (vertical) channel is the number of horizontal (vertical) tracks in the channel. Clearly, there is no terminal inside a horizontal or a vertical channel.

3. Description of the algorithm

In this section, we present the algorithm used in the V4R router that guarantees to use no more than four vias for each net. We first give an overview of the entire algorithm, and then we describe each step of the algorithm in detail.

3.1. Overview of the algorithm

Our algorithm first decomposes each k-terminal net into $k-1$ two-terminal nets based on Prim's minimum spanning tree algorithm. Although the spanning tree topology is used for the initial multi-terminal net decomposition, there are several operations in our algorithm which allow us to introduce Steiner points during the physical routing process so that the final routing solution for each net is a Steiner tree instead of a spanning tree. In the remainder of this section, we assume that each net is a two-terminal net. For each net i, let p_i denote its left terminal (i.e.,

[3] Since the terminals are usually not distributed uniformly in the top layer, the widths of horizontal or vertical channels in each layer may vary significantly. In some MCM technology, several redistribution layers under the top layer are provided for redistributing terminals uniformly before actual routing. After terminal redistribution, the horizontal or vertical channel widths do not vary much. The experimental results in Section 5 are based on the designs without terminal redistribution. We expect that even better results will be achieve if the terminal redistribution technique is applied (at the expense of having extra layers for terminal redistribution).

the one with the smaller column number) and q_i denote its right terminal.

Each net is routed using up to five connected segments alternating between the vertical and horizontal directions. We call a horizontal segment a *h-segment* and a vertical segment a *v-segment*. There are two possible routing topologies depending on the direction of the segment connected to the left terminal or to the right terminal as shown in Fig. 3. The type-1 topology begins with the *left v-stub* from the left terminal, followed by the *left h-segment*, the *main v-segment*, the *right h-segment*, and ending at the right terminal with the *right v-stub*. The type-2 topology begins with the *left h-stub*, followed by the *left v-segment*, the *main h-segment*, the *right v-segment*, and ending with the *right h-stub*. These two types of routing topologies are "orthogonal" to each other. Since each net is routed with no more than five segments, it is clear that each net uses at most four vias.

The reason to consider four-via routing is because it offers sufficient flexibility for connecting a net. In order to connect terminal p located at (0,0) and terminal q located at (m,n), assuming the routing path is within the bounding box of p and q, one-via routing gives two possible routes (i.e. the two L-shape routes), two-via routing gives $(m+n)$ possible routes, three via-routing gives $2 \cdot m \cdot n - (m+n)$ possible routes. Moreover, using no more than three vias allows only monotonic routing paths. However, four-via routing gives roughly $m \cdot n(m+n)$ possible routes, which are usually sufficient in practice. Furthermore, four-via routing allows non-monotonic routing paths.

The V4R router routes two adjacent layers at a time, the odd-number layer is for v-segments and the even-number layer is for h-segments. When routing in the current layer pair, the V4R router maintains a list, named L_{next}, which consists of the nets to be routed in the next layer pair. For each layer pair, it processes column by column starting from the left-hand side. At each column c, the V4R router executes the following four steps:

(1) **Horizontal track assignment of the right terminals**: For each right terminal q_i whose left terminal p_i is in column c, we try to connect q_i to an appropriate horizontal track t_i^r which is

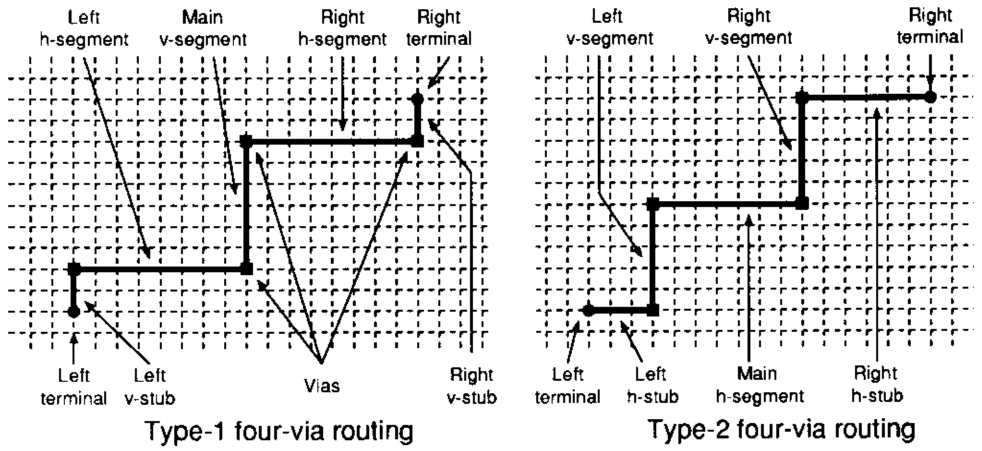

Fig. 3 The two "orthogonal" four-via routing topologies.

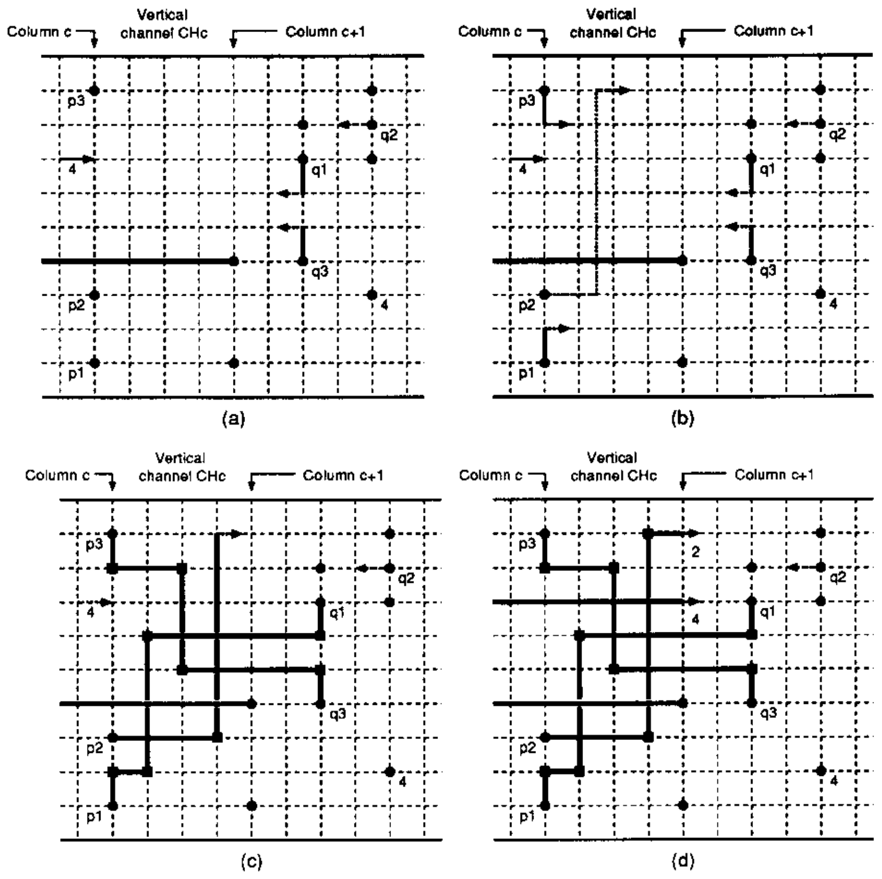

Fig. 4 Overview of the algorithm: the four steps when processing a column c.

free between $col(p_i)$ and $col(q_i)$, using the right v-stub in column $col(q_i)$. The nets that are successfully assigned to feasible tracks are designated as type-1 nets and they will be routed with the type-1 topology. The remaining nets are designated as type-2 nets and they will be routed with the type-2 topology. For example, Fig. 4(a) shows the track assignment for the right terminals q_1 and q_3, which implies that nets 1 and 3 will be routed using the type-1 topology. For a type-1 net i, its right h-segment will be routed in track t_i^r. For a type-2 net j, its right h-stub will be routed in row $row(q_j)$.

(2) **Horizontal track assignment of the left terminals**: There are two phases in this step that process the type-1 and type-2 left terminals independently. In Phase 1, we try to connect each type-1 left terminal p_i in the current column c to an appropriate horizontal track t_i^l using the left v-stub in column c. The left h-segment of a type-1 net i will be routed in track t_i^l. For example, Fig. 4(b) shows the track assignment of the type-1 left terminals p_1 and p_3. In Phase 2, we try to assign a horizontal track for the main h-segment for type-2 left terminals. Note that the main h-segment can be connected to the left terminal only after its left h-stub and left v-segment are routed. Fig. 4(b) shows the track assignment for the type-2 terminal p_2. In both phases, if a terminal p_i cannot be assigned a track, then we rip up all

the routed segments of net i and add i to the list L_{next} to be propagated to the next layer pair.

A net is *active* if its left and right terminals have been assigned the appropriate horizontal tracks yet its routing has not been completed. Clearly, for a type-1 active net, the main v-segment needs to be routed to complete the routing. For a type-2 active net, the left v-segment followed by the right v-segment need to be routed to complete the routing. A v-segment of net i is *pending* if it satisfies any one of the following three conditions: (1) It is the main v-segment of a type-1 active net; (2) It is an unrouted left v-segment of a type-2 active net; (3) It is an unrouted right v-segment of a type-2 active net, its left v-segment has been routed, and the row $row(q_i)$ is free between $col(q_i)$ and column c. The next two steps to be carried out at column c are:

(3) **Routing in the vertical channel**: Select a maximum subset of the pending v-segments and route them in the c-th vertical channel CH_c. Clearly, the density of the selected v-segments should not exceed the capacity of CH_c. In Fig. 4(c), the nets 1, 2, 3 and 4 are all active nets but only the main v-segments of nets 1 and 3 and the left v-segment of net 2 are the pending v-segments and all of them are routed in CH_c.

(4) **Extending to the next column**: We extend the left h-segments of the remaining active nets to column $c + 1$. If the h-segment of an active net i is blocked, we rip up all the routed segments of net i and add i into the list L_{next}. For example, In Fig. 4(d) the h-segments of nets 2 and 4 are extended to column $c + 1$.

After these four steps, we move to column $c + 1$. After we have processed all the columns in the current layer pair, we move to the next layer pair and route the nets in L_{next}. The scanning direction is reversed between the layer pairs to better utilize the routing resources.

It is clear that our algorithm generates detailed routing solutions directly without going through the conventional global routing step. The success of our algorithm depends on the efficient implementation of the above four steps which are carried out at every column. We have developed very efficient algorithms for the four steps based on several combinatorial optimization techniques. The first step is reduced to the problem of finding a maximum weighted bipartite matching, which can be solved in $O(n_c^3)$ time (where n_c is the number of left terminals in column c). Phase 1 of the second step is reduced to the problem of finding a maximum non-crossing matching in a bipartite graph, which can be solved optimally in $O(h_c \log h_c)$ time (where h_c is the number of unoccupied horizontal tracks at current column c). Phase 2 of the second step is again reduced to a maximum weighted bipartite matching, which can be solved in $O(n'^3_c)$ time (where n'_c is the number of type-2 left terminals in column c). The third step is reduced to the problem of finding a maximum weighted k-cofamily in a partially ordered set, which can be solved optimally in $(k_c m_c^2)$ time (where k_c in the capacity of the vertical channel CH_c and m_c is the number of active nets that cross column c). The forth step can be carried out easily. In the remainder of this section, we shall discuss the four steps in detail.

3.2. Horizontal Track Assignment of the Right Terminals

Assume that c is the current column being processed. There are two phases in this step. In the first phase, we determine the terminals for type-1 nets and assign horizontal tracks to them at the same time. In the second phase, we determine the terminals for the type-2 nets.

3.2.1. Phase 1

Let $Q_c = \{q_1, q_2, \cdots q_{n_c}\}$ be the set of the corresponding right terminals of the left terminals in column c. A horizontal track is *feasible* for q_i if (i) it is an unoccupied track and is free between column c and column $col(q_i)$ (excluding columns c and $col(q_i)$), or (ii) it is occupied by a left or right terminal of net i. The objective of this step is to connect each q_i to an appropriate feasible track t_j using the right v-stub so that t_j is reserved for the right h-segment of $net(q_i)$. We build a bipartite graph RG_c in which $q_1, q_2, \cdots q_{n_c}$ are the nodes on the right-hand side and the union of the feasible horizontal tracks of q_i's are on the left-hand side. There is an edge (q_i, t_j) if track t_j is feasible for q_i and we can connect q_i to track t_j using a right v-stub in column $col(q_i)$ without crossing other terminals. For example, Fig. 5(a) shows the right terminals in Q_c and Fig. 5(b) shows the corresponding bipartite graph RG_c.

Moreover, if q_i and q_j are in the same column (say, $y(q_i) < y(q_j)$) and there is no other terminals between q_i and q_j, we only allow q_i to be adjacent to tracks below $\frac{1}{2}(y(q_i) + y(q_j))$ and q_j to be adjacent to tracks above $\frac{1}{2}(y(q_i) + y(q_j))$ in RG_c. (For example, in Fig. 5(b), edges (q_2, t_3) and (q_4, t_2) are not in RG_c although t_3 is feasible for q_2 and t_2 is feasible for q_4.) With this modification of RG_c, it is not difficult to show that any matching in RG_c corresponds to a valid track assignment of the right terminals in Q_c. The nets whose right terminals are

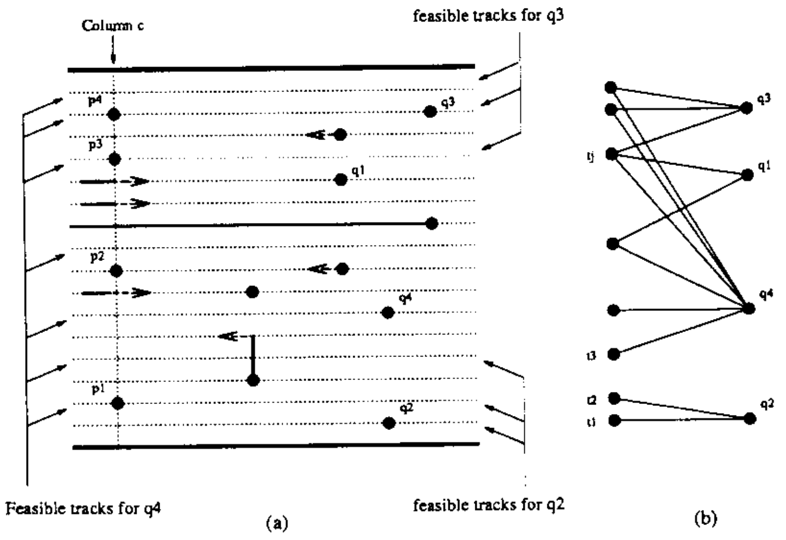

Fig. 5 Construction of the bipartite graph RG_c.

successfully matched in the matching are qualified as type-1 nets, and they will be routed with the type-1 topology.

3.2.2. Phase 2

For the nets whose terminals are not matched in Phase 1, they become the candidates of type-2 nets. These nets will be routed with the type-2 topology.

To optimize the track assignment in Phase 1, we assign an integer weight to each edge (q_i, t_j) in RG_c. Intuitively, if the horizontal track t_j is in the vertical interval formed by $y(p_i)$ and $y(q_i)$ (called the *preferred interval* of q_i), the edge (q_i, t_j) has a large weight (since assigning q_i to t_j in this case minimizes the wirelength). The weight of (q_i, t_j) decreases as t_j moves away from the preferred interval. Hence, our objective in this phase is to find a maximum weighted matching in RG_c.

It is well known that the maximum weighted matching problem in a bipartite graph can be reduced to the minimum cost maximum flow problem in a network which has a cubic time optimal solution (the reader may refer to [FoFu62, Ta83] for the details of the reduction). This result leads to an $O((h'_c + n_c)^3)$ time algorithm for computing a maximum weighted matching in RG_c, where h'_c is the number of the nodes on the left-hand side in RG_c (i.e., the total number of feasible tracks of all the terminals in Q_c) and n_c is the number of nodes on the right-hand side in RG_c (i.e., the number of right terminals in Q_c, which equals to the number of left terminals in column c). Usually, h'_c is much larger than n_c, and a straightforward application of the minimum cost maximum flow algorithm could be inefficient. We show that the bipartite graph RG_c can be simplified as follows. For each node q_i in RG_c, if there are more than n_c edges incident to q_i, we first sort these edges according to the decreasing order of their weights. Then, we keep the first n_c edges (i.e. the edges with large weights) and remove the remaining edges from the graph RG_c. The simplified graph is named RG'_c. Then, we have the following result:

Lemma 1 A maximum weighted matching in the simplified graph RG'_c is also a maximum weighted matching in RG_c.

Proof For each edge (t_i, q_j) in the maximum weighted matching M of RG_c, it must be the case that (t_i, q_j) is one of the n_c highest weighted edges connected to q_i. Otherwise we can always replace (t_i, q_j) by a "free" edges from the n_c highest weighted edges from q_i which is not incident to any matched left-hand node. Therefore, M is also a maximum weighted matching in RG'_c. □

Theorem 1 The horizontal track assignment of the right terminals can be carried out optimally in $O(n_c^3)$ time, where n_c is the number of left terminals in column c.

Proof The optimal track assignment for the right terminals is reduced to finding the minimum cost maximum flow in the simplified bipartite graph RG'_c. Using the algorithm in [Ta83, pp. 110], this can be solved in $O(|M| \cdot m\log_{(2+m/n)}n)$ time where n and m are the number nodes and edges in RG'_c respectively and $|M|$ is the size of the maximum matching. Since the number of edges in RG'_c is at most $n_c \cdot n_c$ (n_c number of edges for each of the n_c right-hand nodes), the optimal track assignment can be solved in $O(n_c^3)$ time. □

This theorem shows that a maximum weighted matching in RG'_c can be computed in $O(n_c^3)$ time, independent of the total number of feasible tracks h'_c. We observed a considerable speed-up when Lemma 1 was applied by the V4R router so that it computes a maximum weighted matching in RG'_c instead of in RG_c.

3.3. Horizontal Track Assignment for the Left Terminals

In this step, we assign the horizontal tracks independently for type-1 and type-2 terminals in Phase 1 and Phase 2, respectively.

3.3.1. Phase 1

Let $P_c = \{p_1, p_2, \cdots p_{n_c}\}$ be the type-1 left terminals in the current column c sorted according to the increasing order of their row numbers. A horizontal track is *unoccupied* if it is not used by any left h-segment of an active net nor is it reserved for any h-segment of an active net. The objective of this step is to connect each type-1 left terminal in P_c to an appropriate unoccupied horizontal track using a left v-stub in column c.

Our algorithm again builds a bipartite graph LG_c in which each node on the left-hand side represents a left terminal p_i in P_c and each node on the right-hand side represents unoccupied horizontal track t_j. There is an edge (p_i, t_j) in LG_c if p_i can be connected to track t_j using a left v-stub in column c without crossing other terminal in the same column. For example, Fig. 6(a) shows the left terminals and the unoccupied tracks at column c, and Fig. 6(b) shows the corresponding bipartite graph. It is clear that a horizontal track assignment of the left terminals corresponds to a matching in LG_c. Moreover, since two v-stubs of two different nets cannot

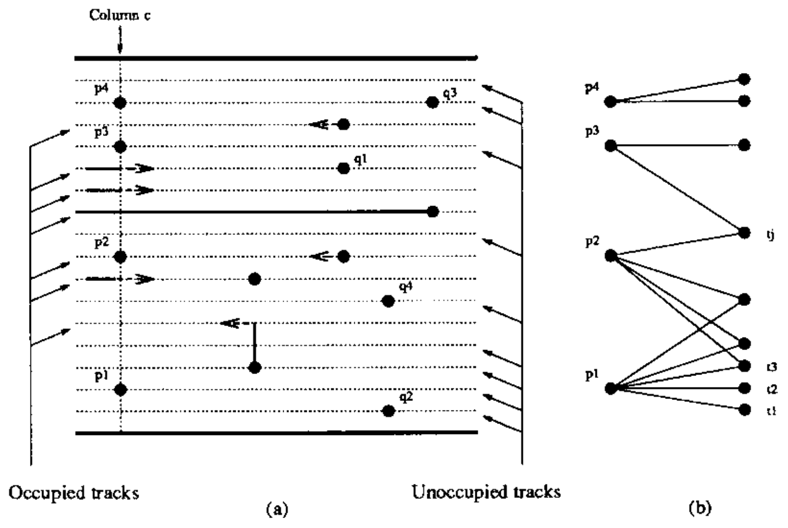

Fig. 6 Construction of the bipartite graph LG_c.

intersect, we require the matching to be *non-crossing*, i.e., there do not exist two edges (p_{i_1}, t_{j_1}) and (p_{i_2}, t_{j_2}) in the matching such that $i_1 < i_2$ but $j_1 > j_2$. Furthermore, each edge (p_i, t_j) may have a weight $w(p_i, t_j)$ which indicates the preference of assigning track t_j to p_i. In general, p_i prefers an unoccupied track which is free between columns c and $col(q_i)$ (or as close to column $col(q_i)$ as possible). Therefore, to find a best track assignment of the left terminals in column c, we want to compute a maximum weighted non-crossing matching in the bipartite graph LG_c. In fact, we want to generalize the definition of a non-crossing matching as follows: if p_{i_1} and p_{i_2} are of the same net, then we allow both (p_{i_1}, t_j) and (p_{i_2}, t_j) to be in a non-crossing matching, i.e., we allow both p_{i_1} and p_{i_2} to be connected to a common track t_j if they are of the same net. (Note that in this case, a Steiner point is introduced in the routing solution of the net.) Consequently, we want to find a *generalized maximum weighted non-crossing matching* in LG_c.

We applied the algorithm by Khoo and Cong presented in [KhCo92] which reduces the problem of computing a generalized maximum weighted non-crossing matching to the problem of computing a maximum weighted chain in the $x-y$ plane. According to their results, we have

Lemma 2 A generalized maximum weighted non-crossing matching in a bipartite graph can be computed in $O(m\log m)$ time, where m is the number of edges in the graph. □

In our case, m corresponds to the number of edges in LG_c. Moreover, we can show that LG_c is not too dense.

Lemma 3 Let h_c be the number of unoccupied horizontal tracks at column c, then the number of edges in LG_c is no more than $2h_c$.

Proof For each right-hand node t_j in LG_c representing an unoccupied horizontal track at column c, there are at most two left-nodes p_i and p_j connected to y_j, where p_i and p_j corresponds to the terminals at column c, which are immediately above or below t_j respectively. Therefore, the number of edges is at most $2h_c$. □

Combing Lemma 2 and Lemma 3, we have

Theorem 2 The horizontal track assignment of the left type-1 terminals in column c can be carried out optimally in $O(h_c\log h_c)$ time, where h_c is the number of unoccupied horizontal tracks at column c. □

3.3.2. Phase 2

Let $P'_c = \{p'_1, p'_2, \cdots p'_{n'_c}\}$ be the type-2 left terminals in the current column c. For a terminal q, let $free\_col(q)$ be the leftmost column such that $row(q)$ is free between columns $free\_col(q)$ and $col(q)$. Then, a *feasible* track for a left terminal p_i is a free horizontal track that does not have any terminal other than those in net i between column c and $free\_col(q_i)$ (excluding column c but including $free\_col(q_i)$) where q_i is the right terminal of p_i. The objective of this phase is to try to assign a feasible track for the main h-segments for each of the type-2 nets. Our algorithm builds a bipartite graph LG'_c in which each node on the left-hand side represents a type-2 left terminal p'_i in P'_c and each node on the right-hand side represents a feasible track for some p'_i. There is an edge (p'_i, t_j) in LG'_c if t_j is a feasible track for p'_i. A weight is assigned to to each edge depending on the length of the free feasible track. A longer free feasible track has a higher weight since the routing in that track is less likely to be blocked. Hence, our objective in this phase is to find a maximum weighted matching in LG'_c. Using a result similar to Lemma 1, we have,

Theorem 3 The horizontal track assignment of the type-2 left terminals can be carried out optimally in $O(n'^3_c)$ time, where n'_c is the number of type-2 left terminals in column c.

Proof The optimal track assignment for the type-2 left terminals is reduced to finding the maximum weighted matching in the bipartite graph LG''_c where LG''_c is obtained by retaining the n'_c highest weighted edges that are incident to each node representing a type-2 left terminal p_i in LG'_c. Using the same arguments as in the proof of Theorem 1, we can show that the optimal track assignment can be solved in $O(n'^3_c)$ time. □

If the left terminal of a net i is not assigned to any horizontal track at the end of this step, we rip up the existing routing segments of net i, and add net i to the list L_{next} to be routed in the next layer pair.

3.4. Routing in a Vertical Channel

Let N_c denote the set of active nets that cross the column c. A v-segment of net $i \in N_c$ is *pending* if it satisfies any one of the following conditions: (1) It is the main v-segment of a type-1 active net; (2) It is an unrouted left v-segment of a type-2 active net; (3) It is the unrouted right v-segment of a type-2 active net, and its left v-segment has been routed, and the row $row(q_i)$ is free between $col(q_i)$ and column c. Moreover, in case (3), we require the endpoints of the pending right v-segment do not share common horizontal tracks with the endpoints of other pending v-segments. (This prevents introducing any vertical constraint in channel CH_c.) The objective of this step is to complete as many active nets in N_c as possible by routing their pending v-segments in the channel CH_c. Note that a type-1 net is completed when its main v-segment is routed and a type-2 net is completed when its right v-segment is routed. For each active net i in N_c, its pending v-segment corresponds to a vertical interval I_i. Let U denote the set of all pending v-segments at column c. It is not difficult to verify that no two endpoints of the pending v-segments share the same horizontal track. Therefore there is no vertical constraint[4] when routing U in channel CH_c. Let $d(S)$ denote the density of the interval set S. Then, we have the following result.

Lemma 4 A set of pending v-segments S can be completed in the channel CH_c if and only if $d(S) \le k$, where k is the capacity of the channel CH_c.

Proof Since there is no vertical constraint among any two pending v-segments, the number of tracks needed to route the v-segments in S is $d(S)$ if we use the left-edge algorithm. Therefore,

[4] The use of the conventional notion of vertical constraint [De76, YoKu82] is a little confusing here since CH_c is a vertical channel. In fact, a vertical constraint here refers to a constraint among the two horizontal segments in the same track that are to be connected to two vertical intervals in CH_c.

the v-segments in S can be completely routed if the channel capacity is at least $d(S)$. □

According to this lemma, to complete as many active nets as possible in CH_c, we want to select a maximum subset S of intervals from U subject to the constraint $d(S) \leq k$. In general, each interval I_i in U may have a positive weight $w(I_i)$ which indicates the priority of the net i to be completed in CH_c. In principle, if $col(q_i)$ is closer to column c, I_i has a higher weight since we want to complete the routing of the net i before we reach its right terminal. We can also assign higher weights to the timing critical nets so that they have a higher chance of being completed in the current layer pair. Therefore, the objective of the vertical channel routing step is to select a maximum weighted subset S of intervals from U subject to the constraint that $d(S) \leq k$.

We shall show that the problem of finding a maximum weighted interval subset $S \subseteq U$ with $d(S) \leq k$ is equivalent to the problem of finding a maximum weighted k-cofamily in a partially ordered set. A *partially ordered set (poset)* P is a collection of elements p together with a binary relation $\leftarrow$ defined on $P \times P$ which satisfies the following three conditions [Li77]:

(1) *reflexive*, i.e., $x \leftarrow x$ for all $x \in P$.
(2) *antisymmetric*, i.e., $x \leftarrow y$ & $y \leftarrow x \Rightarrow x = y$.
(3) *transitive*, i.e., $x \leftarrow y$ & $y \leftarrow z \Rightarrow x \leftarrow z$.

A binary relation satisfying these three conditions is called a *partial ordering relation*. We say that x and y are *related* if $x \leftarrow y$ or $y \leftarrow x$. An *antichain* in P is a subset of elements such that no two elements are related. A *chain* in P is a subset of elements such that every two of them are related. A *k-family* in P is a subset of elements that contains no chain of size $k+1$ [GrKl76]. A *k-cofamily* in P is a subset of elements that contains no antichain of size $k+1$ [GrKl76]. We can have an integer weight $w(p)$ associated with each element p in P. For a subset X of P, the weight of X, denoted $w(X)$, is defined to be the sum of the weights of the elements in X. A *maximum weighted k-family (k-cofamily)* in P is a k-family (k-cofamily) whose weight is maximum. A fundamental result on poset is a theorem due to Dilworth[Di50]:

Theorem 4 (Dilworth, 1950) For a poset P, if the maximum size of antichains is m, then P can be partitioned into m disjoint chains. □

Now we shall define a partial ordering relation on the vertical interval set U. For two vertical intervals $I_1 = (a_1, b_1)$ and $I_2 = (a_2, b_2)$, we say that I_1 is *below* I_2, denoted as $I_1 \leftarrow I_2$, if (i) $b_1 < a_2$; or (ii) $a_1 < a_2$, and $b_1 < b_2$, and I_1 and I_2 segments are of the same net. Moreover, we define $I \leftarrow I$ for any I. For example, in Fig. 7(a), I_8 is below I_4 (according to condition (i)), and I_4 is below I_1 if I_1 and I_4 are of the same net (according to condition (ii)). Intuitively, if I_i is below I_j, then I_i and I_j can be routed in the same vertical track. Allowing two intervals of the same net to overlap is another way of introducing Steiner points.

Lemma 5 The below relation defined on U is a partial ordering relation.

Proof (i) For any interval $I_i = (a_i, b_i)$ in U, by definition we have $I_i \leftarrow I_i$, thus the below relation is reflexive. (ii) Given that $I_1 \leftarrow I_2$ and $I_2 \leftarrow I_1$. If I_1 and I_2 are of different net, then $b_1 < a_2$ and $b_2 < a_1$ which is physically impossible for two different intervals, thus I_1 must be

I_2. If I_1 and I_2 are of the same net, then $a_1 < a_2 < a_1$, which is again impossible for two different intervals, thus I_1 must be I_2. Therefore, the below relation is antisymmetric. (iii) Given that $I_1 \leftarrow I_2$ and $I_2 \leftarrow I_3$, there are four possibilities: (a) I_1 and I_2 are of different nets and I_2 and I_3 are of different nets. Then $b_1 < a_2$ and $b_2 < a_3$. Since $a_2 < b_2$, thus $b_1 < a_3$ and $I_1 \leftarrow I_3$. (b) I_1 and I_2 are of the same net, and I_3 is of a different net. Then $b_2 < a_3$ and $b_1 < b_2$, thus $b_1 < a_3$, which implies $I_1 \leftarrow I_3$. (c) I_2 and I_3 are of the same net and I_1 is of a different net, then $a_2 < a_3$ and $b_1 < a_2$. Thus, $b_1 < a_3$ and $I_1 \leftarrow I_3$. (d) I_1, I_2 and I_3 are all of the same net. Then $a_1 < a_2$, $a_2 < a_3$ and $b_1 < b_2$, $b_2 < b_3$. Therefore, $a_1 < a_3$ and $b_1 < b_3$. Since I_1 and I_3 are of the same net, we have $I_1 \leftarrow I_3$. Therefore, the below relation is transitive. From (i)-(iii), we conclude that the below relation defined on U is a partial ordering relation. □

According to Lemma 5, the vertical interval set U with the below relation forms a poset $P(U)$. For example, Fig. 7(a) shows a set U of vertical intervals. Fig. 7(b) shows the poset $P(U)$ formed by U under the below relation. (We use the Hasse diagram to represent $P(U)$ in which each node represents an interval in U and each edge represents the below relation among a pair of intervals. The edges implied by the transitivity are omitted.) It is not difficult to show that a set of the intervals $S \subseteq U$ can be routed in a single vertical track if and only if S is a chain in $P(U)$. Moreover, we have the following result.

Lemma 6 A set of vertical intervals $S \subseteq U$ (with possibly non-distinct nets) satisfies the constraint $d(S) \le k$ if S is a k-cofamily in $P(U)$.

Proof If S is a k-cofamily, according to Theorem 4 by Dilworth, it can be partitioned into k disjoint chains. Since each chain has density 1, $d(S) \le k$. □

Lemma 7 If a set of vertical intervals $S \subseteq U$ of distinct nets satisfies the constraint $d(S) \le k$, then S is a k-cofamily in $P(U)$.

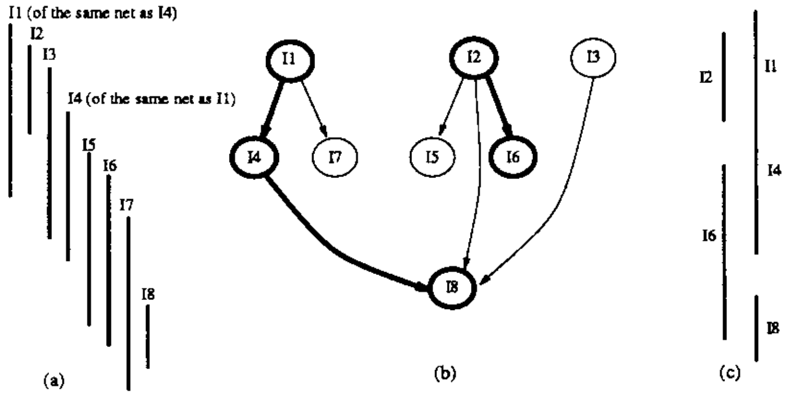

Fig. 7: (a) A set of vertical intervals U where I_1 and I_4 are of the same net.
(b) The partially ordered set $P(U)$ in which the dark edges show a 2-cofamily.
(c) A subset of intervals S with $d(S) \le 2$ induced by the 2-cofamily.

Proof If S is a set of distinct nets with $d(S) \leq k$, we can use the left-edge algorithm to route S into $\leq k$ tracks. Since any two intervals in the same track cannot be in an antichain, we conclude that S has no antichain of size $k + 1$ or larger. □

Note that Lemma 7 may not be true if we have intervals of the same net in S. In a pathological case, suppose we have two intervals of the same net, $I_1 = (a_1, b_1)$ and $I_2 = (a_2, b_2)$, where $a_1 < a_2$ and $b_2 < b_1$. By the definition of the below relation, I_1 is *not* below I_2 because the condition $b_1 < b_2$ does not hold. Therefore, I_1 and I_2 is not a 1-cofamily (i.e. chain) while $d(roman\{I_1, I_2\}) = 1$. However, case like this rarely happens in practice.

According to Lemma 6, the problem of finding a maximum weighted subset of intervals $S \subseteq U$ with $d(S) \leq k$ is reduced to finding a maximum weighted k-cofamily in $P(U)$. For example, the dark edges in Fig. 7(b) shows a 2-cofamily in $P(U)$ and Fig. 7(c) shows the corresponding interval set of density 2. The maximum weighted k-cofamily problem has been studied by Cong and Liu [CoLi90, CoLi91] and by Sarrafzadeh and Lou [SaLo90]. The algorithm by Cong and Liu [CoLi90, CoLi91] computes a maximum weighted k-cofamily in a poset in $O((n-k)n^2)$ time, and the algorithm by Sarrafzadeh and Lou [SaLo90] computes a maximum weighted k-cofamily in a poset in $O(kn^2)$ time, where n is the number of elements in the poset. Both algorithms are based on computing a minimum cost maximum flow in a network. The former is more efficient when k is large (close to n), and later is more efficient when k is small. In our case, the number of tracks in each vertical channel is usually small, so we adopted the algorithm by Sarrafzadeh and Lou in the V4R router. After we obtain a maximum weighted k-cofamily S in $P(U)$, we can route the main v-segments which correspond to the intervals in S in k vertical tracks in channel CH_c and connect each v-segment with the corresponding left and right h-segments of the same net. According to Lemmas 4, 6 and 7, we have

Theorem 5 Routing in the vertical channel CH_c can be carried out in $O(k_c \cdot m_c^2)$ time, where k_c is the capacity of CH_c and m_c is the number of active nets that cross column c. Moreover, when the pending v-segments are of distinct nets, the routing solution is optimal. □

Note that if the main v-segment of a type-1 net is routed in CH_c, the routing of the net is completed. Similarly, if the right v-segment of a type-2 net is routed in CH_c, its routing is also completed.

3.5. Extending to the next column

After we have finished routing in the current vertical channel CH_c, we extend the h-segments of the remaining active nets to the next column $c + 1$. If the h-segment of a net i is blocked by a terminal at column $c + 1$, then we rip up all the routed segments of net i and add i to the list L_{next} to be propagated to the next layer.

After we have process all the columns in the current layer pair, if L_{next} is empty then all the nets have been routed. Otherwise, we propagate the terminals of the nets in L_{next} to the next layer-pair and repeat the routing process on the next layer-pair. The scanning direction for the next layer pair is reversed from the current layer pair so that the routing resource is better utilized.

(In our experiments, if we always scan from left to right, the left side of the routing substrate appears to be more congested.)

3.6. Extensions to the basic algorithm

The algorithms described in the preceding sections are the basis of the V4R router. In this section, we describe three extensions that improve the layer usage and via usage of the router.

The first extension is *back channel routing* of the vertical segments. A back channel is the free vertical space left over from routing in the previous vertical channel. It can be used to route additional pending v-segments that are not routed in the current vertical channel due the channel's capacity constraint. In general, the use of back channels will lead to a slight increase in wirelength and thus it is used only when the routing in the current layer-pair is very congested and when it may help to reduce the number of required layers.

It is possible that the last routing layer pair consists of only a few nets. In this case, we may relax the four-via constraint and re-route the next-to-last layer-pair to accommodate the few nets in the last layer-pair. This mode of routing is called the *multi-via routing* where the h-segments of the remaining active nets at a column can always be extended to the next column using one additional v-segment. The location of the additional v-segment is found efficiently using a simple line scan algorithm. Experimental results show that even with multi-via routing, very few nets use more than 4 vias. In our test examples, the number of nets that used more than 4 vias is less than 7 for any one example and no nets uses more than 6 vias.

The use of orthogonal direction wires between adjacent layers is imposed by our algorithms but seldom by the technology. When the technology allows the use of orthogonal direction wires within a single layer, considerable via reduction may be achieved by moving the v-segments from a v-layer to a h-layer when they do not intersect with any other h-segment or v-segment.

4. Experimental Results

We have implemented the V4R router on Sun workstations using the C language. The V4R router was tested on five examples shown in Table 1. The first three examples labeled test1, test2, and test3 are random examples consisting of only two-terminal nets. The last three examples labeled mcc1, mcc2-75 and mcc2-45 are industrial MCM designs provided by MCC.[5] In particular, mcc2 is a supercomputer with 37 VHSIC gate arrays, and mcc2-75 and mcc-45 are instances of the same design with 75-micron and 45-micron routing pitch, respectively. The experiments reported in this section were performed on a Sun SPARCstation II with 32MB of main memory.

The routing results obtained by the V4R router are shown in Table 2. The second column shows the number of layers used by the V4R router. The third column shows the total number of vias used by the V4R router for each example, which includes both the number of stacked vias used for bringing the terminals to their proper routing layers and the number of vias used for net

[5] We have made these three examples as MCM routing benchmarks for the 4th ACM/SIGDA Physical Design Workshop. These examples are available via anonymous ftp from *mcnc.org* or from the authors *cong@cs.ucla.edu* or *khoo@cs.ucla.edu* directly.

Example	number of chips	number of nets	number of pins	size of substrate (mm^2)	pitch (um)	grid size
test1	4	500	1000	22.5 x 22.5	75	300 x 300
test2	9	956	1912	30 x 30	75	400 x 400
test3	9	1254	2508	37.5 x 37.5	75	500 x 500
mcc1	6	802	2495	45 x 45	75	599 x 599
mcc2-75	37	7118	14659	152.4 x 152.4	75	2032 x 2032
mcc2-45	37	7118	14659	152.4 x 152.4	45	3386 x 3386

Table 1 Test examples.

connection. The forth column shows the average number of vias used for net connection (*excluding* the stacked vias used for bringing the terminals to their routing layers) per two-terminal net. The sixth column shows the total wirelength for each of the routing solutions. We compute a wirelength lower bound for each net i using the formula

$$LB(i) = \max(HP(i), \frac{2}{3}MST(i)) \qquad (1)$$

where $HP(i)$ is the half perimeter of smallest bounding box containing all the terminals in net i, and $MST(i)$ is the wirelength of a minimum spanning tree [6] connecting all the terminals in net i. The wirelength lower bound of each routing example listed in the fifth column is the summation of the wirelength lower bounds of all the nets in the design. The V4R router used at most 4% more wirelength than this lower bound for all examples except for mcc1, which means that the wirelength usage of the V4R router is very close to optimal. (There are many multi-terminal nets in mcc1. Among its 802 nets, 107 of them are multi-terminal nets of size 4 or larger. In this case, the lower bound computed by (1) is considerably smaller than the optimal wirelength. That is why the wirelength for mcc1 by the V4R router is 15% away from the lower bound.)

Table 3 shows the via usage (excluding the stacked vias used to bring the terminals to their routing layers) by the V4R router. For example, 2460 nets or 23.42% of the total number of nets

Example	no. of layers	no. of vias	ave. vias per net	wirelength			run time (hr:min)
				lower bound	V4R	ratio	
test1	4	2250	2.5	102238	104128	1.02	0:01
test2	4	4493	2.7	265000	271067	1.02	0:01
test3	4	5855	2.7	426308	435466	1.02	0:03
mcc1	4	6993	2.2	343767	394272	1.15	0:03
mcc2-75	6	36438	2.9	5362181	5559479	1.04	1:06
mcc2-45	4	36473	2.9	8935372	9130705	1.02	1:37

Table 2 Routing solutions by the V4R router.

[6] It is well known that the wirelength of a minimum spanning tree is no more than 1.5 times that of a minimum Steiner tree in Manhattan routing[Hw76].

Example	Number (percentage) of nets that use						
	0 via	1 via	2 vias	3 vias	4 vias	5 vias	6 vias
test1	8(1.57)	75(12.86)	161(21.64)	171(18.69)	85(8.50)	0	0
test2	10(1.03)	105(9.79)	274(20.36)	353(20.78)	209(10.95)	5(0.26)	1(0.05)
test3	13(1.03)	159(11.15)	367(20.47)	412(18.68)	298(11.91)	4(0.16)	1(0.04)
mcc1	104(5.79)	295(14.10)	695(24.94)	478(14.64)	119(3.52)	0	2(0.06)
mcc2-75	24(0.32)	478(5.94)	2460(23.42)	2323(18.11)	2256(14.96)	0	0
mcc2-45	32(0.42)	586(7.18)	2016(19.81)	2892(22.13)	2014(13.35)	0	1(0.01)

Table 3 Via usage by the V4R router.

in mcc1 use 3 vias each to complete the routing. The examples test2, test3, mcc1, mcc2-45 uses multi-via routing to eliminate very sparse routing layers. In all cases, the number of nets that use more than 4 vias are no more than 6 (0.31%). Furthermore, no more than 6 vias are used for the nets that are routed using multi-via routing.

Table 4 shows the comparison of the V4R router with a general 3D maze router and the SLICE router for multilayer MCM designs [KhCo92]. The 3D maze router failed to produce a routing solution for mcc2-75 and mcc2-75 because of its high memory requirement for large examples. Compared with the 3D maze router, on average the V4R router used 44% fewer vias, 2% less wirelength, ran 26 times faster, and used equal or fewer routing layers. Compared with the SLICE router, on average the V4R router used 9% fewer vias, 2% less wirelength, ran 3.5 times faster and used 1 to 2 fewer routing layers.

A very important advantage of the V4R router is that it does not store the routing grid during the routing process. At any time, the V4R router needs to store only the assignment of the horizontal tracks and the vertical segments of the active nets, which leads to very low memory requirement. For a MCM substrate consisting of K layers of $L \times L$ routing planes, the memory requirement of the V4R router is $\Theta(L+n)$, where n is the number of terminals in the given design. However, the 3D maze router needs to store the entire routing grid, which requires $\Theta(KL^2)$ amount of memory. For large MCM designs, storing the entire routing grid is very expensive or even prohibitive. For the example mcc2-45 (a supercomputer with 37 gate arrays), to store the entire routing grid of size 4×3386×3386, the 3D maze router needs 172MB of memory[7]. That is why the 3D maze router failed to route the example on our system. The SLICE router uses much less memory than the 3D maze router. However, it also needs to store part of the routing grid in order to perform restricted maze routing. The memory requirement of SLICE is $\Theta(\alpha L^2)$, where α is a control parameter (usually takes value between 0.05 and 0.15). If we reduce the pitch spacing by a factor of λ, the memory requirement of both the 3D maze router and the SLICE router increases by a factor of λ^2, while the memory requirement of the V4R router increases by only a factor of λ. Therefore, for the next generation of dense packaging technology, the advantage of the V4R router will become much more significant.

[7] Assume that we use four bytes for each grid point to store the net number, routing cost, etc.

Example	number of layers			number of vias			total wirelength			run time (hr:min)		
	V4R	SLICE	maze	V4R	SLICE	maze	V4R	SLICE	maze	V4R	SLICE	maze
test1	4	5	4	2250	2013	2975	104128	109092	107908	0:01	0:02	0:08
test2	4	6	4	4493	5271	7127	271067	286723	273642	0:01	0:06	0:48
test3	4	6	4	5855	6892	9347	435466	459046	441552	0:03	0:12	1:40
mcc1	4	5	5	6993	6386	8794	394272	402258	397221	0:03	0:12	0:59
mcc2-75	6	7	-	36438	47864	-	5559479	5902818	-	1:06	8:15	-
mcc2-45	4	-	-	36473	-	-	9130705	-	-	1:37	-	-

Table 4 Comparison of the V4R router with the 3D maze router and the SLICE router.

5. Conclusions and Future Extensions

We have presented an efficient multilayer general area router, named V4R, for MCM and dense PCB designs. The unique feature of the V4R router is that it uses no more than four vias to route every net and yet produces high quality routing solutions. It demonstrates elegant applications of several efficient combinatorial optimization techniques to the multilayer general area routing problem. As a result, the V4R router is independent of net ordering, runs much faster, and has far less memory requirement. Compared to the 3D maze router and the SLICE router, the V4R router produced better quality routing solutions in much less computation time.

The cost functions in our graph based algorithms can be tuned to satisfy various performance requirements. For instance, if routing beyond the preferred interval is penalized heavily for the timing critical nets, then the resulting routing for these nets will have shorter wirelength and smaller interconnection delay. Moreover, the vertical tracks within a vertical channel is freely permutable because of the absence of vertical constraint. Therefore, they can be ordered in such a way that the crosstalk between the vertical segments is minimized. Similarly, crosstalk minimization can be taken into consideration when assigning the horizontal tracks of the left and right terminals. The flexibility of incorporating these performance related features makes the V4R router even more attractive for high performance MCM designs.

6. Acknowledgments

We thank Prof. C. K. Cheng at UCSB and Deborah Cobb at MCC for providing the two MCM industrial examples. This research is partially supported by the National Science Foundation under grant MIP-9110511.

7. References

[Ba90] Bakoglu, H. B., *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley Publishing Company, Menlo Park, California (1990).

[Bl83] Blodgett, A. J., ''Microelectronic packaging,'' *Scientific American*, pp. 86-96, July 1983.

[BlBa82] Blodgett, A. J. and D. R. Barbour, ''Thermal conduction module: a high performance multilayer ceramic package,'' *IBM Journal of Research and Development*, Vol. **26**, pp. 30-36, Jan. 1982.

[ChSa91] Cho, J. D. and M. Sarrafzadeh, ''The Pin Redistribution Problem in Multi-Chip Modules,'' *Proc. IEEE ASIC'91*, pp. P9-2.1, 1991.

[CoLi90] Cong, J. and C. L. Liu, ''On the k-Layer Planar Subset and Via Minimization Problems,'' *Proc. of European Design Automation Conf.*, pp. 459-463, March, 1990.

[CoLi91] Cong, J. and C. L. Liu, ''On the k-Layer Planar Subset and Via Minimization Problems,'' *IEEE Trans. on Computer-Aided Design*, pp. 972-981 , Aug. 1991.

[DaDS91] Dai, W. M., T. Dayan, and D. Staepelaere, ''Topological Routing in SURF: Generating a Rubber-Band Sketch,'' *ACM/IEEE 28th Design Automation Conference*, pp. 41-44, 1991.

[DaKJ90] Dai, W. M., R. Kong, J. Jue, and M. Sato, ''Rubber Band Routing and Dynamic Data Representation,'' *IEEE Int'l Conf. on Computer-Aided Design*, pp. 52-55, 1990.

[DaKS91] Dai, W. M., R. Kong, and M. Sato, ''Routability of a Rubber-Band Sketch,'' *ACM/IEEE 28th Design Automation Conference*, pp. 45-48, 1991.

[De76] Deutsch, D. N., ''A Dogleg Channel Router,'' *Proc. 13th Design Automation Conf.*, pp. 425-433, 1976.

[Di50] Dilworth, R. P., ''A Decomposition Theorem for Partially Ordered Set,'' *Ann. of Math*, Vol. **51**, pp. 161-166, 1950.

[FoFu62] Ford, L. R. and D. R. Fulkerson, *Flows in Networks*, Princeton Univ. Press, Princeton, N.J. (1962).

[GrKl76] Greene, C. and D. Kleitman, ''The structure of Sperner k-family,'' *J. Combinatorial Theory, Ser. A*, Vol. **20**, pp. 80-88, 1976.

[HaYY90] Hanafusa, A., Y. Yamashita, and M. Yasuda, ''Three-Dimensional Routing for Multilayer Ceramic Printed Circuit Boards,'' *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pp. 386-389, Nov. 1990.

[He90] Herrell, D., ''Multichip Module Technology At MCC,'' *IEEE Int'l Symposium on Circuits and Systems*, pp. 2099-2103, 1990.

[HoSV90] Ho, J. M., M. Sarrafzadeh, G. Vijayan, and C. K. Wong, ''Layer Assignment for Multichip Modules,'' *IEEE Trans. on Computer-Aided Design*, Vol. **9**, pp. 1272-1277, Dec. 1990.

[Hw76] Hwang, F. K., ''On Steiner Minimal Trees with Rectilinear Distance,'' *SIAM Journal on Applied Mathematics*, Vol. **30**, pp. 104-114, 1976.

[KhCo92] Khoo, K. Y. and J. Cong, ''A Fast Multilayer General Area Router for MCM Designs,'' *EURO-DAC'92*, Sept. 1992. Also to appear in IEEE Trans. on Circuits and Systems, Dec. 1992.

[Li77] Liu, C. L., *Elements of Discrete Mathematics*, McGraw-Hill Book Co., New York (1977).

[Ma84] Marek-Sadowska, M., ''An Unconstrained Topological Via Minimization Problem for Two-Layer Routing,'' *IEEE Trans. Computer-Aided Design*, Vol. **CAD-3**, pp. 184-190, 1984.

[Mi91] Miracky, R. et al, ''Technology for Rapid Prototyping of Multi-Chip Modules,'' *IEEE Int'l Conf. on Computer Design*, pp. 588-591, 1991 .

[PrPC89] Preas, B., M. Pedram, and D. Curry, ''Automatic Layout of Silicon-On-Silicon Hybrid Packages,'' *ACM/IEEE 26th Design Automation Conference*, pp. 394-399, 1989.

[RiKN89] Rim, C. S., T. Kashiwabara, and K. Nakajima, ''Exact Algorithms for Multilayer Topological Via Minimization,'' *IEEE Trans. on CAD*, Vol. **8**, pp. 1165-1173, 1989.

[SaLo90] Sarrafzadeh, M. and R. D. Lou, ''Maximum k-Covering in Transitive Graphs,'' *IEEE Proc. Int'l Sym. on Circuits and Systems*, pp. 332-335, May 1990.

[Sh91] Shambrook, K., ''Overview of Multichip Module Technologies,'' *Multichip Module Workshop*, pp. 1-9, 1991.

[Ta83] Tarjan, R. E., *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania (1983).

[YoKu82] Yoshimura, T. and E. Kuh, ''Efficient Algorithms for Channel Routing,'' *IEEE Trans. on Computer Aided Design of ICAS*, Vol. **CAD-1**, pp. 25-35, Jan. 1982.