

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. It uses no more than four vias to route every net and yet produces high quality routing solutions. 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. As a result, V4R is independent of net ordering, runs much faster, and uses far less memory compared to other multilayer general area routers. Experimental results show that V4R outperforms both the 3D maze router and the SLICE router significantly.

1. Introduction

The multichip module (MCM) technology has been developed recently to increase the packing density and to eliminate a level of interconnection by assembling and connecting bare chips on a common substrate. Due to 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 [BlBa82] and Fujitsu's supercomputer, 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 designs¹.

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.

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 [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.

Several efficient routers have been proposed for silicon-on-silicon based MCM technology [PrPC89, DaDS91]. 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 V4R is that it uses no more than four vias to route every two-terminal net² and yet produces very satisfactory routing solutions. Marek-Sadowska [Ma84] showed a theoretical result that each two-terminal net can be routed using at most one via in a two-layer topological routing solution. Although her result is interesting in theory, the resulting topological solution of 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, V4R 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

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.

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 V4R.

high-performance MCMs, vias not only increase the manufacturing 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 V4R is that it combines global routing and detailed routing in one step and produces high quality *detailed* routing solutions directly from the given net-list 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, V4R is independent of net ordering, runs much faster, and uses far less memory. Experimental results show that V4R outperforms both the 3D maze router and SLICE significantly.

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 conduction vias.

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.

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 degrade the signal's fidelity by introducing impedance discontinuities 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 called a *horizontal track* and a vertical grid line is called a *vertical track*. The terminals on 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*. The channel formed between between the i -th and $(i+1)$ -th rows (columns) is named the i -th horizontal (vertical) channel³. 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 Since the terminals are usually not distributed uniformly in the top layer, the widths of horizontal or vertical channels may vary significantly. In some MCM technology, several redistribution layers under the top layer are

3. Description of the Algorithm

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., 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. 1. 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 $m \cdot n \cdot (m+n)$ possible routes, which are usually sufficient in practice. Furthermore, four-via routing allows

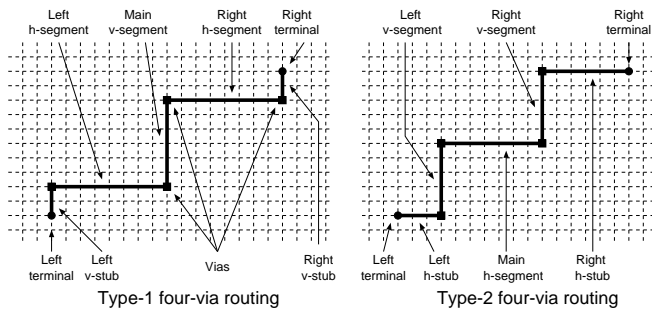


Fig. 1 The two "orthogonal" four-via routing topologies.

provided to redistribute terminals uniformly before actual routing. 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 experimental results in Section 4 are based on the designs without terminal redistribution. We expect even better results if the terminal redistribution technique is applied (at the expense of having extra layers for terminal redistribution).

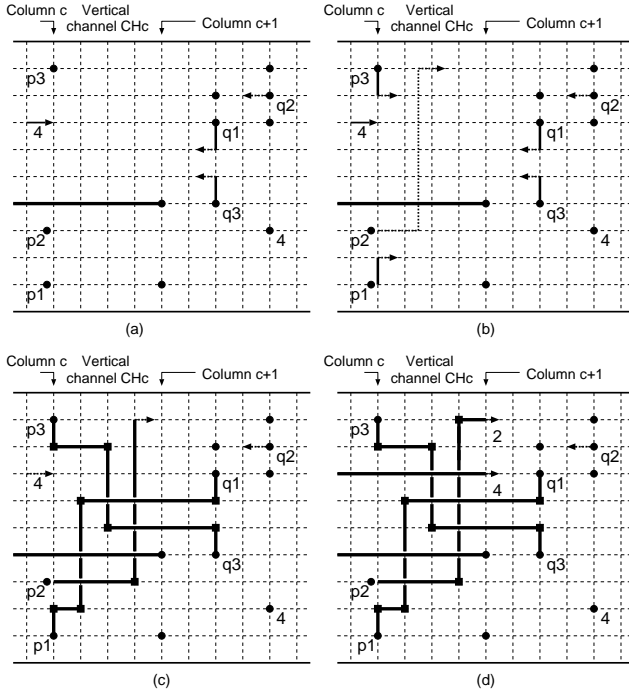


Fig. 2 Overview of the algorithm: the four steps when processing a column c .

non-monotonic routing paths.

V4R 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, V4R 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 columns one by one starting from the left. At each column c , V4R 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 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. 2(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. 2(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. 2(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 . 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 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. 2(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. 2(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 carried out at every column, which will be described in detail in the following subsections. Due to the length restriction, details of these methods and the optimality and complexity analysis of these methods are not presented in this paper. The reader may refer to [KhCo93] for details.

3.2. Horizontal Track Assignment for 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 $Q_c = \{q_1, q_2, \dots, q_{n_c}\}$ 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, \dots, 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. 3(a) shows the right terminals in Q_c and Fig. 3(b) shows the corresponding bipartite graph RG_c .

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. 3(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 . Moreover, each edge (q_i, t_j) may have a weight which indicates the preference of assigning track t_j to q_i . We have shown that any matching in RG_c corresponds to a valid track assignment of the right terminals in Q_c . To optimize the assignment, our algorithm computes a maximum weighted matching in RG_c . Furthermore, we have shown that RG_c can be simplified so that it has at most n_c^2 edges yet still contains a maximum weighted matching, where n_c is the number of left terminals in column c . As a result, the horizontal track assignment of the right terminals can be carried out in $O(n_c^3)$ time. The nets whose right terminals are 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.

3.3. Horizontal Track Assignment for Left Terminals

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

3.3.1. Phase 1

Let $P_c = \{p_1, p_2, \dots, p_{n_c}\}$ be the type-1 left terminals in the current column c sorted according to the increasing order

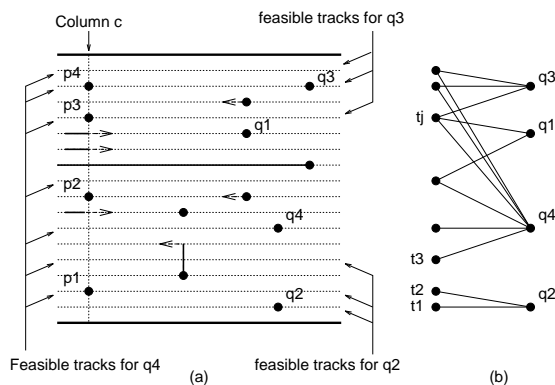


Fig. 3 Construction of the bipartite graph RG_c .

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 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. 4(a) shows the left terminals and the unoccupied tracks at column c , and Fig. 4(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 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 . We have shown that finding a best track assignment of the left terminals is equivalent to computing a maximum generalized weighted non-crossing matching in the bipartite graph LG_c . Moreover, we have shown that the number of edges in LG_c is no more than $2h_c$, where h_c is the number of unoccupied horizontal tracks at column c . Based on these results and the efficient solution to the generalized maximum weighted non-crossing matching problem by Khoo and Cong in [KhCo92], we have concluded that the horizontal track assignment of the left terminals in column c can be carried out optimally in $O(h_c \log h_c)$ time.

3.3.2. Phase 2

Let $P'_c = \{p'_1, p'_2, \dots, p'_{n'_c}\}$ 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 again builds a bipartite graph LG'_c in which each node on the left-

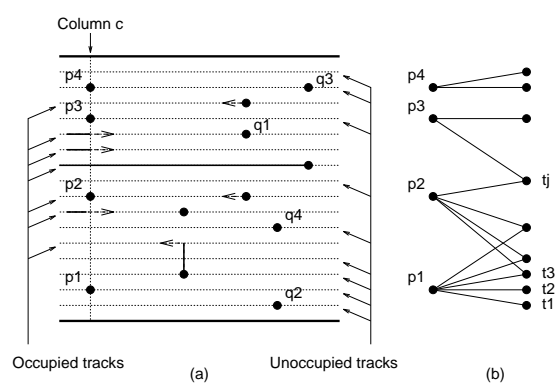


Fig. 4 Construction of the bipartite graph LG_c .

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 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 simplification method similar to that in phase 1 of horizontal track assignment of the right terminals, we conclude that the horizontal track assignment of the type-2 left terminals can be carried out optimally in $O(n_c^3)$ time, where n_c is the number of type-2 left terminals in column c .

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 . 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 . Each interval I_i may have a positive weight $w(I_i)$ which indicates the priority of the net i to be completed in CH_c . Let $INT(N_c)$ denote the set of vertical intervals $\{I_i | i \in N_c\}$. We define a partial ordering relation on $INT(N_c)$ as follows. 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. 5(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. It is not difficult to show that $INT(N_c)$ under the below relation forms a partially ordered set (poset). A k -cofamily in a poset is the union of no more than k chains [GrK176, CoLi91]. We have shown that computing the optimal routing of the main v-segments in channel CH_c is equivalent to computing a maximum weighted k_c -cofamily in $INT(N_c)$, where k_c is the capacity of CH_c . Based on these results and efficient algorithms for computing a maximum weighted k -cofamily [CoLi91, SaLo90], we have concluded

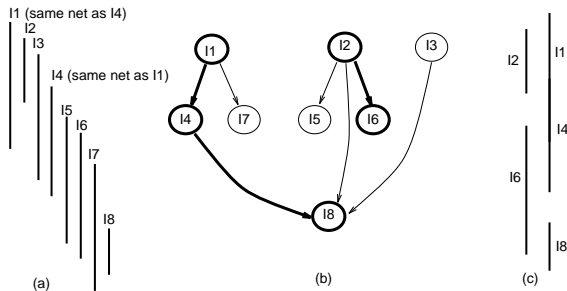


Fig. 5: (a) A set of vertical intervals $INT(N_c)$ where I_1 and I_4 are of the same net. (b) The poset formed by $INT(N_c)$ in which the dark edges show a 2-cofamily. (c) A subset of intervals S with $d(S) \leq 2$ induced by the 2-cofamily.

that routing in the vertical channel CH_c can be carried out optimally in $O(k_c \cdot m_c^2)$ time, where m_c is the number of active nets that cross column c .

3.5. 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 and via usage of the router.

The first extension is *back channels routing* of the vertical segments. If the current column is c , CH_k ($k < c$) are the back channels which may have some free vertical space for routing additional pending v-segments that are not selected in the current vertical channel. 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 is called *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 determined efficiently using a simple line scan algorithm. However, the use of multi-via routing is highly restricted. In every test example, no more than 7 nets are routed using multi-via routing and none of them 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 V4R on Sun workstations using the C language and tested it on six examples shown in Table 1. The first three examples 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.⁴ In particular, mcc2 is a supercomputer with 37 VHSIC gate arrays, and mcc2-75 and mcc2-45 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.

Example	No. of chips	No. of nets	No. of pins	Size of substrate (mm^2)	Grid size
test1	4	500	1000	22.5×22.5	300×300
test2	9	956	1912	30×30	400×400
test3	9	1254	2508	37.5×37.5	500×500
mcc1	6	802	2495	45×45	599×599
mcc2-75	37	7118	14659	152.4×152.4	2032×2032
mcc2-45	37	7118	14659	152.4×152.4	3386×3386

Table 1 Test examples. The routing pitch is $75\mu m$ for all the examples except mcc2-45 which has a routing pitch of $45\mu m$.

⁴ 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 layers			Number of vias			Total wirelength			Run time (hr:min)			
	V4R	SLICE	Maze	V4R	SLICE	Maze	Lower bound	V4R	SLICE	Maze	V4R	SLICE	Maze
test1	4	5	4	2250	2013	2975	102238	104128	109092	107908	0:01	0:02	0:08
test2	4	6	4	4493	5271	7127	265000	271067	286723	273642	0:01	0:06	0:48
test3	4	6	4	5855	6892	9347	426308	435466	459046	441552	0:03	0:12	1:40
mcc1	4	5	5	6993	6386	8794	343767	394272	402258	397221	0:03	0:12	0:59
mcc2-75	6	7	-	36438	47864	-	5362181	5559479	5902818	-	1:06	8:15	-
mcc2-45	4	-	-	36473	-	-	8935372	9130705	-	-	1:37	-	-

Table 2 Comparison of the V4R router with the 3D maze router and the SLICE router.

Table 2 shows the comparison of V4R with a general 3D maze router and SLICE for multilayer MCM designs. 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 V4R used 44% fewer vias, 2% less wirelength, ran 26 times faster, and used equal or fewer routing layers. Compared with SLICE, on average V4R used 9% fewer vias, 2% less wirelength, ran 3.5 times faster and used 1 to 2 fewer routing layers. Moreover, V4R used at most 4% more wirelength than the lower bound⁵ for all examples except mcc1⁶, which means that the wirelength usage of V4R is very close to optimal.

A very important advantage of V4R is that it does not store the routing grid during the routing process. It stores 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 V4R 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 and SLICE needs to store a portion of the two-layer routing grid, which requires $\Theta(\alpha L^2)$ amount of memory (α is a parameter usually 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 V4R increases by only a factor of λ . Therefore, for the next generation of dense packaging technology, the advantage of V4R will become much more significant.

5. Future Extensions

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

⁵ The lower bound of the wirelength for each net i is computed using $LB(i) = \max(HP(i), \frac{2}{3}MST(i))$ 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 connecting all the terminals in net i . (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].)

⁶ 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 is considerably smaller than the optimal wirelength. That is why the wirelength for mcc1 by V4R is 15% away from the lower bound.

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 V4R 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).
- [BiBa82] 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.
- [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.
- [GrK176] 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.
- [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 in *IEEE Trans. on Circuits and Systems*, Nov. 1992.
- [KhCo93] Khoo, K.-Y. and J. Cong, "Four Vias Are Sufficient In Multilayer MCM and Dense PCB Routing," *UCLA Computer Science Department Tech. Report CSD-930001*, Los Angeles, CA, Jan. 1993.
- [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.
- [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.