

A Fast Four-Via Multilayer MCM Router

Kei-Yong Khoo and Jason Cong

Department of Computer Science
University of California at Los Angeles
Los Angeles, CA 90024, U. S. A.

Abstract

In this paper, we present an efficient multilayer general area router, named V4, for MCM and dense PCB designs. The unique feature of the V4 router is that it uses no more than four vias to route every net and yet produces high quality routing solutions. A number of combinatorial optimization techniques are used in the V4 router to produce high quality routing solutions in polynomial time. As a result, the V4 router is independent of net ordering, runs much faster, and has far less memory requirement compared to other multilayer general area routers. We tested our router on several examples, including two industrial MCM designs from MCC. Compared with the 3D maze router, on average the V4 router uses 2% less wirelength, 31% fewer vias, and runs 26 times faster. Compared with the SLICE router, on average the V4 router uses 4% less wirelength and runs 4.6 times faster.

1. Introduction

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. 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 pitch 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 despite the fact that there are many interconnection layers. 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 a number of 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 31% 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, DaKJ90, DaDS91, DaKS91]. Since the number of signal routing layers is usually small in this technology, some 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 V4, for MCM and dense PCB designs. The unique feature of the V4 router is that it uses no more than four vias to route every two-terminal net² and yet produces very satisfactory routing solutions. This result

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. 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 V4 router.

makes an interesting contrast with the theoretical result by Marek-Sadowska [Ma84] which shows that each two-terminal net can be routed using at most one via in a two-layer topological routing solution. Although the result in [Ma84] is very interesting in theory, the resulting topological routing solution 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 V4 router is the first practical multilayer general area router which guarantees to use a small 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].

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 of the modules are brought to the first signal routing layer through distribution vias. The goal of MCM routing is to connect the I/O terminals in each net 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].

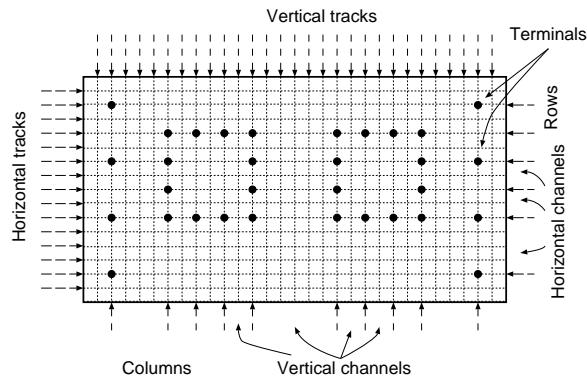


Fig. 1 Illustration of the definitions.

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.

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 the same horizontal track form a *row*, and the terminals on the same vertical track form a *column*. For a terminal x , $row(x)$ and $col(x)$ denote the row number and the column number of x respectively. Two adjacent rows form a *horizontal channel*, and two adjacent columns form a *vertical channel*. (See Fig. 1.) The channel formed between the i -th and $(i+1)$ -th rows (columns) is named the i -th horizontal (vertical) channel. Clearly, there is no terminal in a horizontal or a vertical channel.

3. Description of the algorithm

In this section, we present the algorithm used in the V4 router which 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., the one with the smaller column number) and q_i denote its right terminal.

The V4 router routes two adjacent layers at a time, one layer for horizontal wires and the other layer for vertical wires. When routing in the current layer pair, the V4

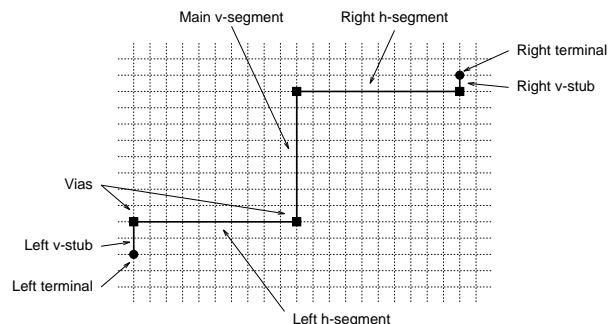


Fig. 2 Each net is routed using at most four vias.

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 V4 router executes the following four steps:

- (1) **Horizontal track assignment of the left terminals:** For each left terminal p_i in column c , we connect it to an appropriate horizontal track t_i^l using a vertical segment in column c called the *left v-stub* of net i . The horizontal segment to be routed in track t_i^l is called the *left h-segment* of net i (see Fig. 2). For example, Fig. 3(a) shows the track assignment of the left terminals p_1, p_2, p_3 , and p_4 at column c .
- (2) **Horizontal track assignment of the right terminals:** For each right terminal q_i whose left terminal p_i is in column c , we connect it to an appropriate horizontal track t_i^r , which is free between $col(p_i)$ and $col(q_i)$, using a vertical segment in column

$col(q_i)$ called the *right v-stub* of net i . The horizontal segment to be routed in track t_i^r is called the *right h-segment* of net i (see Fig. 2). For example, Fig. 3(b) shows the track assignment of the right terminals q_1, q_2, q_3 , and q_4 . If we fail to assign q_i to a feasible horizontal track, we rip up the left v-stub of net i and add i to the list L_{next} .

A net is *active* if we have assigned its left and right terminals to the appropriate tracks yet its routing has not been completed. Clearly, for an active net i , we need to use a vertical segment to connect the two horizontal tracks t_i^l and t_i^r . Such a vertical segment is called the *main v-segment* of net i (see Fig. 2). The next two steps to be carried out at column c are:

- (3) **Routing in the vertical channel:** Select a maximum subset of the main v-segments of all the active nets and route them in the c -th vertical channel CH_c . Clearly, the density of the selected main

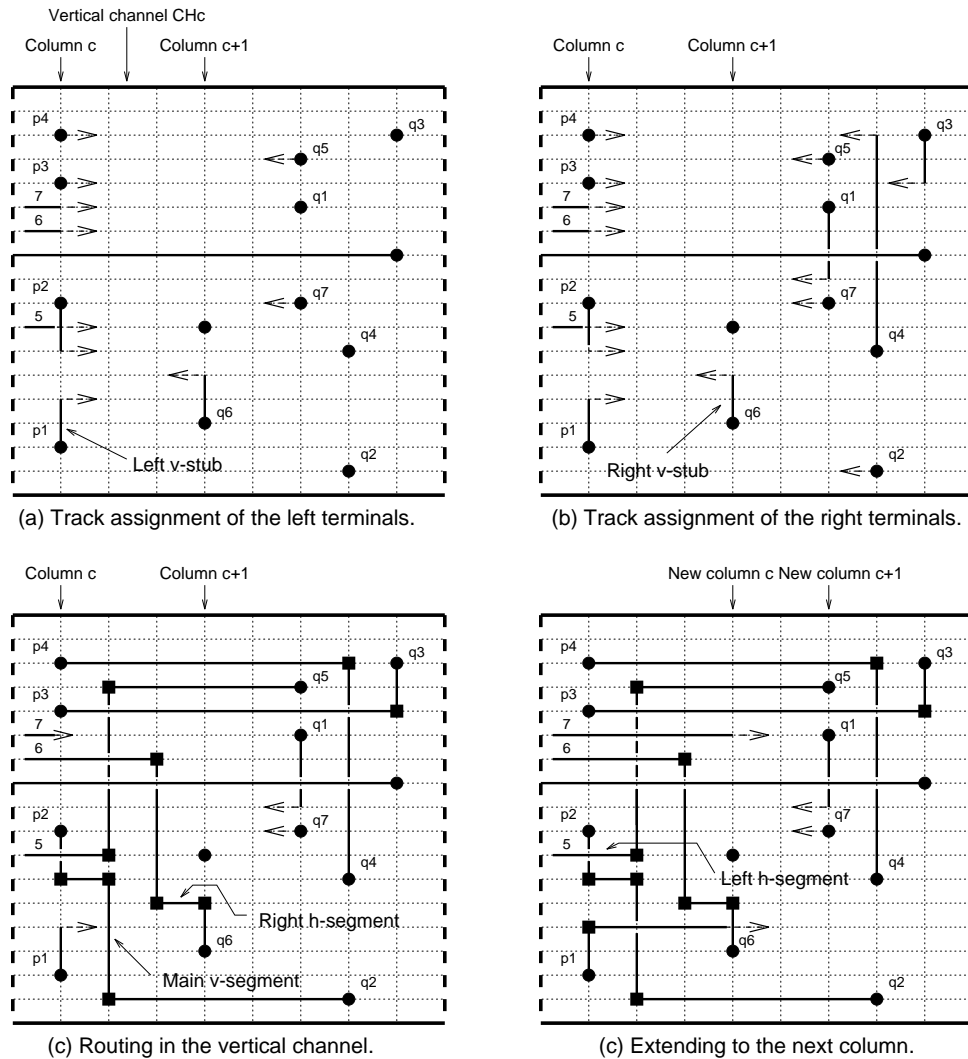


Fig. 3 Overview of the algorithm.

v-segments should not exceed the capacity of CH_c . In Fig. 3(c), the nets 1 to 7 are active nets and the nets 2, 3, 4, 5, and 6 are completed in CH_c (note that the v-segments of nets 3 and 4 are of zero length).

- (4) **Extending to the next column:** We extend the left h-segments of the remaining active nets to column $c + 1$. If the left h-segment of an active net i is blocked, we rip up the left h-segment, the left v-stub, and the right v-stub of net i and add i into the list L_{next} . The nets in L_{next} will be routed in the next layer pair. For example, In Fig. 3(d) the left h-segments of nets 1 and 7 are extended to column $c + 1$.

After these four steps, we move to column $c + 1$. It is clear from the description of our algorithm that each net is routed using at most three vertical segments (the left v-stub, the main v-segment, and the right v-stub) and at most two horizontal segments (the left h-segment and the right h-segment). Therefore, each net is routed using at most four vias. (We do not count the stacked vias for bringing the terminals to the proper routing layer, since these vias have to be used independent of the choice of routing algorithms.)

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} . In our implementation, we rotate the routing substrate by 90 degree before we move to the next layer pair so that the scanning direction for the next layer pair is orthogonal to that for the current layer pair. Such an implementation leads to a better utilization of the routing resources.

In the remainder of this section, we describe the methods used for the first three steps of the algorithm (the fourth step is straightforward). 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 the Left Terminals

Let $P_c = \{p_1, p_2, \dots, p_{n_c}\}$ be the 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 right h-segment of an active net. The objective of this step is to connect each terminal in P_c to an appropriate unoccupied horizontal track using a left v-stub in column c . Our algorithm constructs 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 an 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. Moreover, 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 generalized maximum weighted non-crossing matching in the bipartite graph LG_c . Moreover, we have shown that LG_c is not too dense. More precisely, 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 $2h_c$. 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. Horizontal Track Assignment of the Right Terminals

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 a (right) v-stub so that t_j is reserved for the right h-segment of *net* (q_i) . We construct 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 (t_j) 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. 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

$\frac{1}{2}(y(q_i) + y(q_j))$. Moreover, each edge (q_i, t_j) may have a weight $w(q_i, t_j)$ 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 . Based on these results and the minimum cost maximum flow algorithm by Tarjan[Ta83], we have concluded that the horizontal track assignment of the right terminals can be carried out in $O(n_c^3 \log n_c)$ time.

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 main v-segments in the channel CH_c . For each active net i in N_c , its main v-segment corresponds to a vertical interval I_i . Each interval I_i in $INT(N_c)$ may have a positive weight $w(I_i)$ which indicates the priority of the net i being completed in CH_c . Let $INT(N_c)$ denote the set of vertical intervals $\{I_i | i \in N_c\}$. We have shown that $INT(N_c)$ forms a partially order set. Moreover, 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 . (For the definition of a *k-cofamily*, see [GrK176, CoLi91].) Based on these results and the efficient algorithm for computing a maximum weighted *k-cofamily* by Sarrafzadeh and Lou [SaLo90], we have concluded 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 .

4. Experimental Results

We implemented the V4 router on Sun workstations using the C language. The V4 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 two examples labeled mcc1 and mcc2 are industrial MCM designs provided by MCC. In particular, the example mcc2 is a supercomputer with 37 VHSIC gate arrays. The routing pitch is 75 μm for all the examples. The experiments reported in this section were performed on a Sun SPARCstation II with 32MB of main memory.

The routing results obtained by the V4 router are shown in Table 2. The second column shows the number of layers used by the V4 router. The third column shows the total number of vias used by the V4 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 connection. The fourth column shows the total wirelength for each of the routing solutions. We compute a wirelength lower bound for each net i using the formula

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

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 tree³ connecting all the terminals in net i . The wirelength lower bound of each routing example listed in the fourth column is the summation of the wirelength lower bounds of all the nets in the design. The V4 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 V4 router is very close to optimal. (Since there are many multi-terminal nets in mcc1, the lower bound computed by Eq. (1) is no longer tight.)

Table 3 shows the distribution of the completed nets as the number of routing layers increases. It shows that 57-77% of the nets are routed in the first layer pair. For all test examples, no more than 5% of the nets are routed in the last layer pair. This indicates that it is very likely that we can reroute the nets in the last layer pair in the previous layer pairs in a postprocessing step so that we may reduce the number of routing layers for most of the

Ex.	# of chips	# of nets	# of pins	size of substrate (mm^2)	grid size
test1	4	500	1000	22.5 x 22.5	300 x 300
test2	9	956	1912	30 x 30	400 x 400
test3	9	1254	2508	37.5 x 37.5	500 x 500
mcc1	6	802	2495	45 x 45	599 x 599
mcc2	37	7118	14570	152.4 x 152.4	2032 x 2032

Table 1 Characteristics of test examples.

³ 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].

Ex.	# of layers	# of vias	wire length			run time (hr:min)
			lower bound	V4	ratio	
test1	4	2501	102238	104654	1.03	0:01
test2	6	4914	265000	270627	1.02	0:01
test3	6	6413	426308	436250	1.02	0:03
mcc1	6	7804	343767	388938	1.13	0:04
mcc2	8	38541	5516890	574038	1.04	1:00

Table 2 Routing solutions by the V4 router.

examples by two. Currently, we are working on such an enhancement.

Table 4 shows the comparison of the V4 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 due to its high memory requirement for large examples. Compared with the 3D maze router, on average the V4 router used 2% less wirelength, 31% fewer vias, and ran 26 times faster. Compared with the SLICE router, on average the V4 router used the same number of vias but 4% less wirelength, and ran 4.6 times faster. The V4 router used slightly more routing layers than the 3D maze router and the SLICE router due to the restriction of the routing topology generated by the V4 router in order to guarantee four-via routing. However, as discussed in the preceding paragraph, there is a good chance that we can reduce the number of routing layers used by the V4 router by two for most examples.

A very important advantage of the V4 router is that it does not store the routing grid during the routing process. At any time, the V4 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 plane, the memory requirement of the V4 router is $\Theta(L + n)$, where n is the number of terminals in the given design. However, the 3D maze router requires $\Theta(KL^2)$ amount of memory, and the SLICE router requires $\Theta(\alpha L^2)$ amount of memory, where α is a control parameter determining the range of maze router. If we reduce the pitch spacing by a factor of k , the memory requirement of both the 3D maze router and the SLICE router increases by a factor of k^2 , while the memory requirement of the V4 router increases by only a factor of k . Therefore, for

Example	% of nets completed in x layers			
	1-2	3-4	5-6	7-8
test1	77.2	100.0		
test2	57.6	95.1	100.0	
test3	58.1	96.2	100.0	
mcc1	74.3	99.5	100.0	
mcc2	56.66	90.6	99.8	100.0

Table 3 Distribution of the completed nets in different layer pairs.

Example	number of layers			number of vias			total wire length			run time (hr:min)		
	V4	SLICE	maze	V4	SLICE	maze	V4	SLICE	maze	V4	SLICE	maze
test1	4	5	4	2501	2013	2975	104654	109092	107908	0:01	0:02	0:08
test2	6	6	4	4914	5271	7127	270627	286723	273642	0:01	0:06	0:48
test3	6	6	4	6413	6892	9347	436250	459046	441552	0:03	0:12	1:40
mcc1	6	5	5	7804	6386	8794	388938	402258	397221	0:04	0:12	0:59
mcc2	8	7	-	38541	47864	-	5740438	5902818	-	1:00	8:15	-

Table 4 Comparison of the V4 router with the 3D maze router and the SLICE router.

the next generation dense packing technology, the advantage of the V4 router will become much more significant.

5. Conclusions

We have presented an efficient multilayer general area router, named V4, for MCM and dense PCB designs. The unique feature of the V4 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 V4 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 V4 router produced better quality routing solutions in much less computation time.

6. Acknowledgments

The authors thank Prof. C. K. Cheng at UCSD 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).
- [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.
- [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.

- [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.
- [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,
- [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.
- [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).