

A Fast Multilayer General Area Router for MCM Designs

Kei-Yong Khoo and Jason Cong
Department of Computer Science
University of California at Los Angeles
Los Angeles, CA 90024, U. S. A.

Abstract

The objective of this research is to develop an efficient multilayer general area router as an alternative to the three-dimensional (3D) maze router for solving the multilayer MCM routing problem. Our router, named SLICE, is independent of net ordering, requires much shorter computation time, and uses fewer vias. A key step in our router is to compute a maximum non-crossing bipartite matching, which is solved optimally in $O(n \log n)$ time where n is the number of possible connections. We tested our router on a number of examples, including two MCM designs from MCC. The total wirelength used by SLICE is only a few percent away from the optimal. Compared with a 3D maze router, SLICE is four times faster and uses 28% fewer vias. A more important feature is that SLICE works on only a "thin slice" of the two-layer routing grids at a time, while a 3D maze router works on the entire three dimensional routing grid. Therefore, SLICE can successfully produce solutions for large MCM routing examples where 3D maze routers fail due to insufficient memory.

1. Introduction

In order to reduce interconnection delay and increase packaging density, the multichip module (MCM) technology is used in the design of high-performance VLSI systems. 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 (including 1 bonding layer, 5 distribution layers, 16 interconnection layers, 8 voltage reference layers, and 3 power distribution layers [BIBa82, B183]). Fujitsu's latest super-computer, 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 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.

¹ Beside the problem of efficient utilization of routing resource, there are also a number of 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.

Few methods are available for 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 a number of 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. For example, one industrial example that we obtained from MCC has a 75 micron routing pitch and a routing area of $174 \times 174 \text{ mm}^2$; this results in a routing grid of 2032×2032 for a single layer! It is certainly not a trivial task to store such a grid for each layer and search in it efficiently.

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, it faces a few problems. First we have to pre-determine the number of the routing layers before we can carry out layer assignment. Moreover, the approach does not take advantage of the large number of routing layers. Thus, each net is often forced to use many vias since it has to be routed within two layers. For high-performance MCMs, vias not only increase the manufacture cost but also degrade the system performance since they form inductive and capacitive discontinuities and cause signal reflections[Ba90].

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 in this technology, some techniques for IC routing, such as hierarchical routing and rubber-band routing, can be applied to yield good solutions.

The objective of our research is to develop an efficient multilayer general area router as an alternative to the 3D maze router for solving the multilayer MCMs routing problem. Our router, named SLICE, has a number of advantages. First, it processes many nets simultaneously so that the routing solution is independent of net ordering. Moreover, it requires much shorter computation time and much smaller memory storage. Finally, it emphasizes planar routing so that most of the nets use very few vias. A key step in our method is to compute a maximum non-crossing bipartite matching, which is solved optimally in $O(n \log n)$ time (where n is the number of possible connections). We tested our router on a number of examples, including two MCM designs from MCC, and compared

the results with those by a 3D maze router. On the average, both routers use about the same amount of wires, but the 3D maze router is 4 times slower and uses 28% more vias. A more important feature is that SLICE works on only a "thin slice" of the two-layer routing grids at a time, while a 3D maze router works on the entire three dimensional routing grid. Therefore, SLICE can successfully produce solutions for large MCM routing examples where 3D maze routers fail due to the memory requirement.

The remainder of this paper is organized as follows. Section 2 formulates the multilayer MCM routing problem. In Section 3, we give an overview of our algorithm and we describe each step in the algorithm in detail. Experimental results and a comparative study are presented in Section 4. Finally, we discuss the extension of our work in Section 5.

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 routing grid superimposed on each routing layer where the spacing between grid lines is determined by the routing pitch for the given P/I technology. We assume that the routing grid is a Manhattan grid. However, our algorithm can handle 45 degree routing as well. 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].

The output of the routing problem is a set of wire segments and vias that connect each net. 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 due to impedance discontinuity in signal paths thus should also be minimized. Vias usually cause more serious problems than jogs, so that our router gives via minimization a higher priority. Each additional routing layer increases the manufacturing cost. Thus, the number of layers should also be minimized.

3. Description of the algorithm

In this section, we present our fast multi-layer general area router, called SLICE, for MCMs and single-sided PCBs designs. We first give an overview of the entire algorithm, and later we describe each step in more detail.

3.1. Overview of the algorithm

The basic idea behind our algorithm is to perform planar routing on a layer by layer basis. After routing on one layer, we propagate the terminals of the uncompleted nets to the next layer. Then we continue routing on the next layer and perform the single layer routing again. We repeat the process until all the nets are routed.

A crucial part of our algorithm is to compute a planar routing solution for each layer and try to connect as many nets as possible in that layer. For nets that cannot be completed in the layer, we try to perform a partial routing so that these nets can be completed in the next layer with shorter wires. We scan the routing region from left to right and process each pair of columns that has terminals at a time. For each adjacent column-pair, we compute a maximum weighted non-crossing matching (MWNCM) which consists of a set of non-crossing edges that extend from the left column to the right column. This gives us a topological planar routing solution between the column-pair. Next, we generate a physical routing between the current column-pair based on the selected edges in the non-crossing matching. Then, we move on to the next column-pair and compute the non-crossing matching and physical routing again for that column pair. The planar routing process is completed when the right end of the current layer is reached. Clearly, such a left to right scanning operation results in mainly horizontal wires in the planar routing solution. To complete the routing in the vertical direction, we use a restricted two-layer maze router that is much faster than a general maze router. We clean up the routing solution by removing unnecessary jogs and wires in the current layer. For nets that are not completed, their terminals are propagated to the next layer. Finally, we rotate the substrate by 90 degrees so that the scanning direction in the next layer is orthogonal to the one used in the current layer which helps to complete vertical connections. The entire process is iterated until all the nets are routed. We shall describe each step in detail in the remaining subsections.

3.2. Planar routing

The terminals that lie on the same vertical grid line form a *column*. In our planar routing algorithm, we scan the routing region across from left to right and perform routing between each column-pair. Let x_l be the x -coordinate of the left column, and x_r be the x -coordinate of the right column. Conceptually, a column-pair forms a channel and we define the channel capacity to be $C_{cap} = x_r - x_l$. During planar routing in the current layer, the terminals of the uncompleted nets are put in the list P_{prop} . These terminals will be propagated to the next routing layer.

For each column pair, the occupied grid points on the left column are called *start-points*. Each start-point is either a terminal propagated from the previous layer, or the endpoint of a partial routing solution computed in the previous channel. We denote a start-point n_i in the current layer by a triple $n_i = (x_i, y_i, net_i)$, where (x_i, y_i) is the coordinates of the point, and net_i is the net number of the point. For a start-point n_i , the terminal that it is to be connected to is called the *target* of n_i , denoted by

$target(n_i)$.²

We shall concentrate our discussion on routing between a single column pair. We begin with a list P_l that contains all the start-points on the left column. After that, we go through three steps to complete the planar routing. (1) For each start-point, we generate a set S of weighted edges that connect the start-point to the right column. The weight for each edge represent the gain if we include this connection in the planar routing solution. (2) We compute the maximum weighted non-crossing matching S_{MWNCM} of S , which corresponds to the best topological planar routing solution between the column pair. (3) Finally, we compute the physical routing solution based on the edges in S_{MWNCM} . The steps in the planar routing algorithm are illustrated in Figure 1. We now describe these three steps in detail.

3.2.1. Generating the weighted edges

Given a list of start-points P_l on the left column, we want to generate the set S of weighted edges that connects the start-points in P_l to the grid points on the right column. Conceptually, for a start-point $n_i = (x_i, y_i, net_i)$ in P_l , we can generate an edge from (x_i, y_i) to every grid point on the right column which is free or occupied by a terminal of net n_i . However, this may result in too many edges, and most of them will have very little chance of being selected in the maximum non-crossing matching in the next step. To conserve both memory and time, we use a simple heuristic, called *range reduction*, to reduce the number of edges that are generated. Clearly, the channel capacity C_{cap} bounds the density of the vertical segments that can be routed between a column-pair. Let y_{i+n} and y_{i-n} be the y -coordinate of the n -th start-point (not grid point) above and below n_i on the left column respectively. Now if we assume that all the start-points on the left column will be routed, then the connection for each start-point above or below n_i will increase the channel density by one. Therefore, it is sufficient to generate edges whose right endpoints are within the y -interval $[y_i - C_{cap}, y_i + C_{cap}]$. This is the reduced range where the edges for n_i can end on the right column.

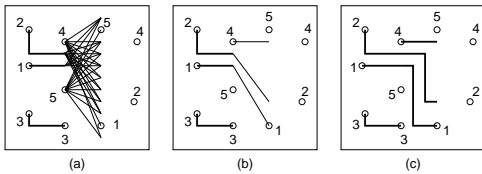


Figure 1: Steps in planar routing for a column-pair. (a) Generate possible (topological) connections; (b) Compute a maximum weighted non-crossing matching; (c) Generate physical routing.

² As a preprocessing step of SLICE, we decompose each multi-terminal net into a set of two-terminal subnets based on the Prim's minimum spanning tree algorithm. Therefore, each start-point always has a well-defined target. The prior decomposition does not affect the routing quality very much since (1) most nets are two-terminal nets in MCM routing; (2) we allow the routes of two subnets of the same net to meet and form a Steiner point in our planar routing procedure.

The weight of each edge represents the gain if we include the edge in our planar routing solution. For each start-point n_i , let $n_j = target(n_i)$. We define the *preferred region* to be the y -interval on the right column defined by the y -coordinates of n_i and n_j . Clearly, if an edge from n_i ends within the preferred region, it does not increase the wirelength of net_i . So we assign a high weight, $weight_preferred$, to the edges ending in the preferred region. Moreover, if an edge from n_i ends exactly on n_j (when n_j is on the right column), we assign an even higher weight, $weight_complete$, to the edge in favor of completion of the net. For the edges that end outside the preferred region, we assign them a small weight $weight_outside$. In general, $weight_complete > weight_preferred > weight_outside$. We experimented with several weighting functions and the best choice is the following. We set $weight_complete$ and $weight_preferred$ to be two constants, and let $weight_outside$ decrease linearly as the right end of the edge moves away from the preferred region.

3.2.2. Computing the MWNCM

The most important part of our planar routing routing algorithm is computing a topological planar routing solution between each column-pair. We begin with a set of weighted edges $S = \{s_1, s_2, \dots, s_n\}$. Each edge in S represents a possible topological route that extends from a start-point to the right column. The weight for each edge represents the gain if we include this route in the planar routing solution. Each edge s_i is a four-tuple (l, r, w, net) where l is the y -coordinate of the left end of the edge, r is the y -coordinate of the right end of the edge, w is the weight of the edge, and net is the net number of the edge. Since we want a set of best edges that are planar so that they can be routed on the current layer, we need to select a set of edges from S that are non-crossing and have the maximum total weight. This is the *maximum weighted non-crossing bipartite matching (MWNCM) problem*. However, in our formulation, we permit two edges in a non-crossing matching to share a common endpoint at the right column if they belong to the same net. (In this case, a steiner point is generated.)

In order to compute a MWNCM, we map each edge to a point in the x - y plane using the one-to-one mapping $(l, r) \rightarrow (x, y)$ where (x, y) is the position of the point in the x - y plane. Given two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ in the x - y plane, we say that p_i dominates p_j if (i) $x_i > x_j$ and $y_i > y_j$, or (ii) $x_i > x_j$ and $y_i = y_j$ and

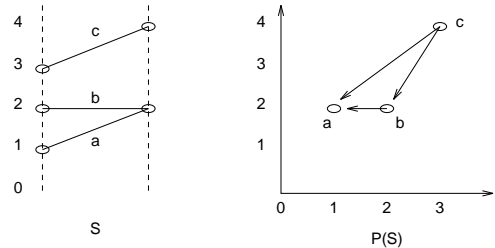


Figure 2: Mapping the edges to points in the plane. Edges in $P(S)$ indicate the dominance relations (assuming the start-points 1 and 2 are of the same net).

$net_i = net_j$ (net_i is the net that the corresponding edge of p_i belongs to). The dominance relations are illustrated in Figure 2, where edge c dominates edges a and b , and edge b dominates edge a (assuming that a and b are of the same net). Clearly, if p_i dominates p_j , the two edges that are mapped to p_i and p_j are either strictly non-crossing or sharing the same endpoint on the right column if they are of the same net. We define a *chain* among a set of points P in the x - y plane, to be an ordered list of points $C = \{p_1, p_2, \dots, p_m\}$ where $p_k \in P$, and p_{k+1} dominates p_k for $k = 1, 2, \dots, m-1$. We define the *weight* of a chain C , denoted by $weight(C)$, to be the sum of the weights of the points in C . We define the *maximum-chain* C_{max} to be the chain that has the maximum weight among a given set of points. We then have the following results:

Lemma 1: The dominance relation is transitive.

Theorem 1: Let S be the set of edges between a column pair and $P(S)$ be the set of corresponding points on the x - y plane. Then the set of edges M in S is a maximum weighted non-crossing matching if and only if the corresponding points $P(M)$ form a maximum-chain in $P(S)$.

The proofs of the results in this section can be found in [KhCo92]. A maximum-chain of a point set P can be computed as follows: We construct a direct graph G_P , called the *dominance graph*, in which each node represents a point in P . We add an edge (p_i, p_j) to G_P if and only if point p_i dominates point p_j . Figure 3 shows a set of edges and the dominance graph on the corresponding point set. (the edges implied by the transitive relation are omitted for clarity.) It is not difficult to show that G_P thus constructed is a directed acyclic graph and a maximum-chain in P corresponds to a maximum weighted path in G_P . Since the maximum weighted path in a directed acyclic graph can be computed in $O(n^2)$ time, where n is the number of nodes in the graph [ReND77], we can compute a MWNCM in $O(n^2)$ time, where n is the number of edges we generated between a column-pair. The maximum weighted edges in S and the maximum weighted path in $P(S)$ are shown in Figure 3 as dark edges. However, since the procedure for computing a maximum weighted non-crossing matching will be used for each column-pair, we seek for a more efficient implementation. We have developed an $O(n \log n)$ time algorithm for computing the MWNCM based on a data structure called the priority search tree [Ta83]. The details of

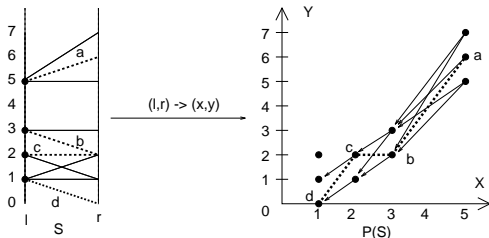


Figure 3: A set of edges S and its corresponding point set $P(S)$. Note that start-points 2 and 3 are of the same net. The dominance relations are shown in $P(S)$ as thin arrows. The MWNCM in S , and the corresponding maximum-chain in $P(S)$ are shown as dotted arrows.

the algorithm is given in [KhCo92].

Theorem 2: Given the set S of n edges between a column pair, the maximum weighted non-crossing matching can be computed in $O(n \log n)$ time.

We noticed a significant speed-up when the $O(n^2)$ time algorithm was replaced by the $O(n \log n)$ time algorithm for computing a maximum weighted non-crossing matching between each column pair.

3.2.3. Physical routing

The solution we obtained from computing the maximum weighted non-crossing matching in the previous section gives us a set of edges, $S_{MWNCM} = \{s_1, s_2, \dots, s_n\}$, where $s_i = (l_i, r_i, net_i)$, l_i and r_i are the y -coordinates of the left and right endpoints of the edge, and net_i is the net number of the edge. Each edge represents a connection between the two points, (x_l, l_i) and (x_r, r_i) in the current routing plane, where x_l and x_r are the x -coordinates of the left and right columns, respectively. As the result of the physical routing, we end up with a list of endpoints of the routings, P_r , on the right column, together with the terminals that are on the right column, will form the new list of start-points on the new left column in the next iteration of column-pair routing. Because the edges in S_{MWNCM} are topological routing solutions, not all edges can be routed due to the channel capacity constraint. We add to P_{prop} the start-points whose edges failed to be routed.

We perform the routing separately on two classes of edges from S_{MWNCM} as defined below. Given an edge $s_i = (l_i, r_i, net_i)$, we say that s_i is a rising edge if $l_i < r_i$. Otherwise, we say that s_i is a falling edge (i.e. $l_i \geq r_i$). We group all the rising edges in S_{rise} and all the falling edges in S_{fall} . We also order the edges in S_{fall} in *increasing* y -coordinates, and order the edges in S_{rise} in *decreasing* y -coordinates. It is not difficult to show that the edges in S_{rise} and S_{fall} can be routed separately.

We perform the physical routing one edge at a time. We shall describe the routing for S_{rise} . (Routing for S_{fall} is similar.) For each edge s_i in S_{rise} , we start routing from (x_l, l_i) in the routing plane, and route towards (x_r, r_i) in the following manner. We advance the routing along the y -axis upward until the routing is blocked, or if we have reached the y -coordinate r_i . Then we shall route one grid unit along the x -axis rightward if possible and repeat the routing along the y -axis. This process is repeated until one of the following three cases is encountered. (1) The routing is completed, (2) the routing has ended on the right column but did not reach (x_r, r_i) , and (3) the routing has failed to reach the right column. For case (1) where the routing is completed, we simply add the start-point

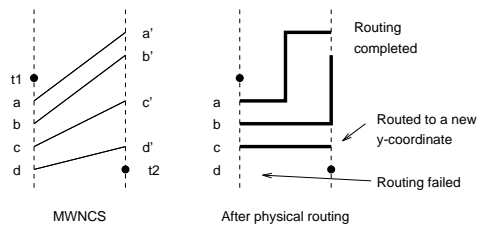


Figure 4: Example of physical routing.

(x_r, r_i, net_i) to P_r . For case (2), we add the start-point with the new y-coordinate, (x_r, new_r_i, net_i) to P_r where r_{new_i} is the y-coordinate where the routing ends on the right column. For case (3) we remove any partial routing that we might have added between the column-pair, and add the start-points $n_i = (x_l, l_i, net_i)$ and $target(n_i)$ to the list of terminals P_{prop} to be propagated to the next layer. Figure 4 illustrates these cases. The left side of Figure 4 shows four rising edges in the MWNCM named $a, b, c,$ and d . Terminals $t1$ and $t2$ are of other nets. Routing begins with edge a and b which are completed. Edge c is routed to the right column but at a different y-coordinate. Finally we fail to route edge d because of the blocking terminal on the right column. In this case, both the start-point of the edge d and its target will be added to P_{prop} .

3.3. Restricted maze routing

Our planar algorithm will produce routing segments extending predominantly in the scanning direction. Therefore, many start-points may not be routed because they are lined up almost vertically. To complement the planar routing, we use a restricted maze router to complete as many left over nets as possible.

To conserve memory, we restrict the maze-routing to within two routing layers. Moreover, we restrict the range of the maze router to a thin "vertical slice" of the routing region since we are primarily interested in vertical connections. Typically, the maze range is 10% of the width of the routing region.

3.4. Jog removal

Since the planar routing algorithm does not penalize the formation of wiring jogs, the completed routings may contain many unnecessary jogs. Therefore, a clean up phase is necessary to remove these jogs to improve the quality of the planar routing solution.

We identify two kinds of jogs. We call *simple jogs* to be those that can be eliminated by moving a single wire segment. Otherwise, the remaining jogs are called the *complex jogs*, where more than one wire segments need to be moved to eliminate a jog. SLICE tries to remove the simple jogs first, then it tries to remove the remaining complex jogs. Both algorithms are based on the efficient plane sweeping technique used extensively in computational geometry [PrSh85]. Experimental results show that on the average, 39% of the simple jogs, and 37% of the complex jogs can be removed by our algorithms.

4. Experimental Results

We implemented SLICE on the Sun workstations using the C language. The following experimental results were recorded on a Sun SPARC station II with 32MB of memory. We tested the program on five examples shown in Table 1. The examples test1, test2, and test3, are generated with random netlists. Examples mcc1 and mcc2 were industrial MCM routing examples provided by MCC. Example mcc2 is a supercomputer with 37 VHSIC gate arrays.

The routing results obtained by SLICE on these examples are given in Table 2. The lower bound on wire length for each net is computed by the half perimeter of the bounding rectangle that encloses all the terminals in the net. This is a conservative lower bound for multi-terminal

nets. It can be seen that SLICE uses at most 7% more than this lower bound for all examples except mcc1³.

Ex.	no. of chips	no. of nets	no. 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 examples (pitch = $75\mu m$)

Ex.	no. of layers	no. of vias $\times 100$	no. of jogs $\times 100$	wire length ($\times 1000$)			run time hr:min
				lower bound	SLICE	ratio	
test1	6	22	39	102	109	1.065	0:04
test2	7	56	100	265	282	1.064	0:11
test3	7	76	148	426	455	1.067	0:18
mcc1	6	69	124	339	399	1.177	0:18
mcc2	9	516	1111	5623	5821	1.035	7:36

Table 2: Characteristics of solutions

Table 3 shows that a large percentage of the nets are completed within the first few routing layers. For all cases, more than 74% of the nets are completed within the first 4 routing layers.

Ex.	% of nets completed in x layers								
	1	2	3	4	5	6	7	8	9
test1	5.2	55	82	94	99.4	100			
test2	2.8	31	59	82	95.6	99.4	100		
test3	2.6	32	58	81	95.1	99.1	100		
mcc1	2.0	29	63	84	99.1	100			
mcc2	1.3	32	57	74	87.9	94.4	98.4	99.7	100

Table 3: Distribution of completed nets

Table 4 shows the effect of the two jog removal algorithms. Note that each of the individual jog removal algorithms may remove both kind of jogs, thus the total jogs removed by applying both algorithm is less than the sum of the jogs removed by applying the two individual algorithms independently.

We also compare our results with a general 3D maze router in Table 5. The 3D maze router uses a reserved layer model^{4,5}, in which the horizontal wires and vertical

³ There are many multi-terminal nets in mcm1. Since the lower bound given by the half perimeter of the bounding box is not tight for multi-terminal nets of size 4, our lower bound for mcm1 is considerably smaller than the optimal wirelength. In fact, it is commonly believed that the wirelength of a minimum Steiner tree for a multi-terminal net is at most 88% of the wirelength of a minimum spanning tree on average [BeCa87], which leads to a new wirelength lower bound of 362497 for mcm1. The result by the SLICE router is only 10% more than the new lower bound.

⁴ The 3D maze router using the reserved layer model performed much better than the one using the un-reserved layer model. For example, in our experiment, the number of layers required for example mcc1 is 17 with the un-reserved layer model versus 5 layer with the reserved layer model.

⁵ We tried both the reserved layer and the un-reserved layer model for the two-layer maze router in SLICE, but the difference in the results is insignificant. For all test cases, the results by SLICE reported in Table 2-5 are based on the two-layer un-reserved layer model.

wires are routed in different layers. The 3D maze router was not able to run on mcc2 on our system due to the large size of the example. On the average, SLICE is more than four times faster than the 3D maze router, and uses 28% fewer vias than the 3D maze router. However, the number of layers used by SLICE is generally more than the 3D maze router. But as shown in Table 3, the last few layers in the SLICE solutions are very sparse. In fact, the uncompleted nets on the last layer is less than 0.6% for all the test examples. Therefore, it is possible to manually reroute the nets on the last or last few layers so that the total number of layers is reduced.

Ex.	Total number of jogs			
	Without jog removal	Algo. to remove Simple Jogs	Algo. to remove Complex Jogs	Both Algorithms
test1	7213	4190	4489	3879
test2	19366	11485	11686	9986
test3	27846	17185	17623	14805
mcc1	21700	13240	14079	12405

Table 4: Effects of jog removal algorithms

Ex.	no. of layers		no. of vias ($\times 100$)		no. of jogs ($\times 100$)		wire length ($\times 1000$)		run time (hr:min)	
	S	M	S	M	S	M	S	M	S	M
test1	6	4	22	30	39	4	109	108	0:04	0:08
test2	7	4	56	71	100	9	282	274	0:11	0:48
test3	7	4	76	93	148	11	455	442	0:18	1:40
mcc1	6	5	69	88	124	12	397	397	0:18	0:59
mcc2	9	-	516	-	1111	-	5821	-	7:36	-

Table 5: Comparison with maze router (S = SLICE, M = maze router)

A significant advantage that SLICE has over the 3D maze router is its low memory requirement. For the example mcc2 (a supercomputer with 37 gate arrays), in order to store the entire grid of size $9 \times 2032 \times 2032$, the 3D maze router needs 148MB of memory (assuming that we use four bytes for each grid point to store the net number, routing cost, etc.) That is why the 3D maze router failed to route the example on our system. However, using a maze routing range of 10%, at any time, the working space of SLICE is only $2 \times 10\% \times 2032 \times 2032 = 3.3\text{MB}$ of memory. So SLICE successfully produced a satisfactory solution. Furthermore, if the routing pitch for the same example is halved, the 3D maze router will require 595MB of memory whereas SLICE will require only 13.2MB of working memory. Clearly, for the next generation of dense MCM design, the 3D maze router will face more severe memory limitation, and the advantage of SLICE will become much more significant.

5. Conclusions and Future Extensions

In this paper, we present a fast multilayer general area router named SLICE for MCM Designs. The routing result of SLICE router is independent of net ordering and uses fewer vias. The total wirelength produced by SLICE is only a few percents away from the optimal. Compared with a general 3D maze router, with a small penalty in the number of routing layers, SLICE runs about four times faster, uses 28% fewer vias, and requires far less memory.

The SLICE router can also handle 45 degree routing. After we obtain a maximum weighted non-crossing matching, we can use a more sophisticated procedure to

map the topological routing solution into a physical routing solution which allows 45 degree routing. The SLICE can handle arbitrary obstacle in the routing region as well since it can avoid generating edges whose end-points are on the obstacles. We hope to take some performance issues into consideration in the design of next generation of SLICE.

6. Acknowledgments

The authors thank Prof. C. K. Cheng and his colleagues 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).
- [BeCa87] Bern, M. W. and M. d. Carvalho, "A Greedy Heuristic for the Rectilinear Steiner Tree Problem," in *UC Berkeley Computer Science Department Tech. Report 87-306*, Berkeley, CA (1987).
- [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.
- [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.
- [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.
- [KhCo92] Khoo, K.-Y. and J. Cong, "A Fast Multilayer General Area Router for MCM Designs," in *UCLA Computer Science Department Tech. Report CSD-920011*, Los Angeles, CA 90024 (March, 1992).
- [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.
- [ReND77] Reingold, E., J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1977).
- [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).