A New Timing-Driven Multilayer MCM/IC Routing Algorithm

Dongsheng Wang Ernest S. Kuh

Dept. of Electrical Engineering and Computer Sciences
University of California
Berkeley, CA 94720, USA

Abstract

In high-performance multilayer routing, time delay is an important performance issue which has not been appropriately addressed by previous multilayer routing approaches. This paper proposes a new timing-driven MCM/IC multilayer routing algorithm, named MLR, considering the Elmore delay as well as some other fundamental performance issues, such as the number of layers, vias and the total wirelength. Algorithm MLR assigns all the nets into the routing layers layer-pair by later-pair based on the layer assignment algorithm. During each pair-layer routing, the timing-driven Steiner area routing algorithm SOAR is used to generate a Steiner tree for each net while minimizing the Elmore delay of the net. For two nodes to be connected for the net being routed, an optimal path from one node to the other is created by the $(\alpha, \beta)^*$ algorithm. Additionally, when power and ground nets are considered, some signal nets are routed in the limited routing space on the power and ground layer-pair, which is very useful in decreasing the number of layers needed to complete the routing. The proposed algorithm has been implemented and tested on CBL/NCSU and MCC benchmarks and the experimental results are very promising.

1 Introduction

The development of VLSI fabrication technology and the advent of deep sub-micron technologies have made multilayer routing problems dominant in performance-driven physical design for high density MCMs and ICs.

There exist some multilayer routing approaches [1]-[7]. One of the approaches is four-via routing algorithm V4R in [4]. V4R routes two adjacent layers (i.e., a layer-pair) at one time. The odd-number layers is for vertical segments and the even-number layer is for horizontal segments. For each layer-pair, V4R processes columns one by one from left to right. But V4R's column-by-column method may introduce more vias in the large-size grid routing. The approach in [3] divides the routing layers into several layer-pairs. Nets are assigned to these layer-pairs and then a two-layer routing is carried out for each layer-pair. But this approach needs to pre-determine the number of routing layers before the layer assignment is carried out. Generally speaking, the pre-determination is hard to make.

SLICE in [5] and M^2R in [2] are another kind of multilayer routers. SLICE computes the routing solution on a layer-by-layer basis and carry out planar routing in each layer. However, since planar routing can complete only a limited number of nets, a two-layer maze router is used to complete the remaining nets, which slows down the computation and introduces extra vias. The strategy of M^2R router is to do single layer routing first for more critical nets, followed by layer-pair routing for less critical nets. That is, it first iterates single layer routing until $\alpha\%$ of the nets have been routed, then route other $(100 - \alpha)\%$ nets by applying layer-pair routing iteratively.

Another significant approach in [6] produces multilayer rubber-band sketches. It uses two phases, global routing and local routing, to route nets layer by layer. Although this approach can also deal with multilayer routing, the generalization of the technique to multilayer general area routing is not clear.

The MCG algorithm in [1] attempts to achieve the highest possible routing density on any given routing layer-pair. It considers a small number of possible routes for each net and constructs a compatibility graph. Then this compatibility graph is reduced to yield a subset of routes which are fully compatible. In addition, a three-phase routing strategy is used to route nets with as few vias as possible, which can efficiently decrease the number of vias.

Recently, a challenging 3-D multilayer MCM routing approach is introduced in [7]. It can efficiently utilize the three dimensional routing space by decomposing the complex three dimensional routing problem into a set of simpler tower routing problems. But this approach routes the residual nets by using 3-D maze router, which may lead to lower routing efficiency and high running time consumption.

However, some limitations exist in these existing approaches. First, few of them have accurately estimated time delay which is very important for high performance multilayer MCM/IC design. The approaches in [2] shows the lower bound of the time delay based on the lower bound of wirelength and the number of vias. However this estimation is not accurate because the time delay of interconnection depends not only on the wirelength and vias, but also on the net topology. The approach in [2] estimates the delay only in the initial global routing phase. No final time delay was available. Another important issue on multilayer routing is to minimize the number of vias used. Some approaches try to use fewer vias to complete routing, such as the

0-8186-7789-9/97 $10.00 © 1997 IEEE

approaches in [4, 2, 1]. But the decrease in the number of vias is very limited. Lastly, For the sake of simplicity, most multilayer routing approaches by and large split the multi-terminal net into two-terminal subnets based on the minimal Spanning tree method, and then route them separatively. This may lose the net topological information, and influence the final routing performances.

In this paper, we propose a new timing-driven multilayer routing algorithm, named MLR, for both IC and MCM designs. Following our earlier multi-objective optimization placement algorithm in [9] and performance-driven global routing algorithm in [10], MLR algorithm further solves the detailed routing problem to complete the whole physical design for high performance MCMs and ICs. The unique feature of MLR is that it can accurately estimate the Elmore delay for each net by maintaining the completed net topologies during the whole routing process. In addition, MLR pursues some other performance issues, such as wirelength, the number of routing layers and vias, and tries to optimize these routing performances. Experiments show that MLR outperforms MCG, V4R and SLICE singnificantly. Furthermore, the crosstalk, a parasitic coupling (mutual capacitances and inductances) phenomenon between neighboring signal nets, is also a very important performance issue for multilayer MCM/IC routing. An appropriate crosstalk constraint should be imposed on the routing problem, and a "crosstalk-free" routing solution should be obtained. This issue will be discussed in detail in another papers [11] and [12].

2 Problem Formulation

We assume that there is a Manhattan routing grid imposed on each routing layer where the space between grid lines is determined by the routing pitch for the given technology. We complete the interconnection of all the nets based on the routing grid, and each net is connected by some horizontal and vertical segments, called h-segments and v-segments respectively. An example of the four-layer routing for a four-pin net is shown in Fig. 1. MLR routes two adjacent layers, called the layer-pair, at a time. The odd-number layer is for h-segments and the even-number layer is for v-segments. Two-pin nets are connected in the same layer-pair and multi-pin nets may be connected in the same or different layer-pairs. Two segments in same layer-pair may be connected by a simple via. Two segments in different layer-pair can be connected by a stacked via.

The input of our multilayer routing problem includes 1) a given routing grid whose size is determined by the routing pitch. 2) a set of interconnection nets whose pins are located on the routing grid, and each net may have either two or more pins. 3) timing parameters including unit resistance, unit capacitance, driver resistances of source pins and load capacitances of sink pins. These parameters are necessary for computing Elmore delay of each sink pin.

The output of the problem is a set of routing segments and vias that connect all the nets on multiple routing layers. The quality of the routing are measured by the number of layers required to complete the routing, the number of vias, the total wirelength and the time delay. The more the number of layers is, the more the manufacturing costs. Therefore the number of layers should be minimized. Long wires increase propagation time and should be avoided. Vias and bends degrade the signal's fidelity by introducing impedance discontinuities in signal path thus should be also minimized. Time delay is a very important measurement of routing performances, therefore it should accurately estimated. We use Elmore delay model [?] to estimate the maximum sink delay in our algorithm.

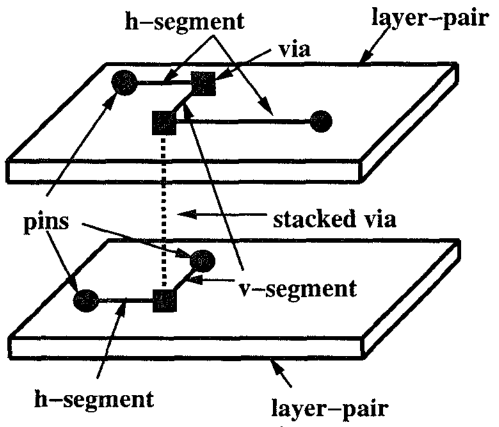
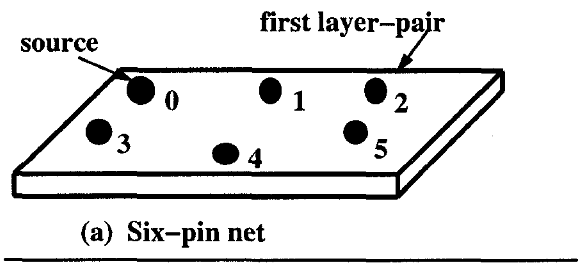


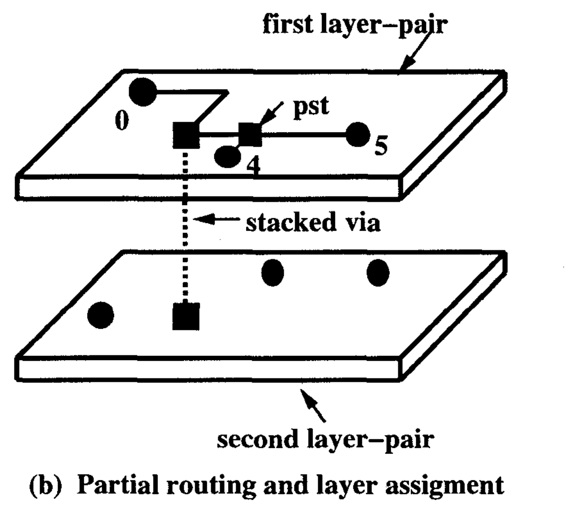
Fig. 1. Four-layer routing for a four-pin net

3 Multiple Layer Routing Algorithm

3.1 Overview of the Algorithm

The goal of Multiple Layer Routing (MLR) algorithm described in this paper is to determine the final detailed routes for all nets with high performances. MLR algorithm mainly consists of layer assignment (LA) algorithm, Steiner optimal area routing (SOAR) algorithm and hierarchical $(\alpha, \beta)^\star$ routing algorithm. LA algorithm, based on layer-pair routing, exactly assigns each net into particular layers and determines the exact position of each net segment on a particular layer. It uses a separative layer-pair to route power and ground nets and the limited space available on this layer-pair is used to route some signal nets. This strategy is useful in saving routing space for other signal net routing layers. SOAR algorithm routes the multi-pin nets to minimize the maximum sink delay. It handles each multi-pin net as an entirety instead of decomposing it into a set of 2-pin nets. This enables us to obtain the completed net topology which is useful to compute the time delay. SOAR algorithm also allows the multi-pin nets to be routed in different layer-pairs. All the 2-pin nets are routed by $(\alpha, \beta)^\star$. $(\alpha, \beta)^\star$ algorithm, based on the ideas of (α, β) algorithm in [8], finds a optimal routing path between two given pins using fewer vias and minimum path length on current layer-pair. Being different from (α, β) algorithm, $(\alpha, \beta)^\star$ algorithm has two unique features. One is its hierarchical mechanism. Although (α, β) algorithm has a powerful routing ability, it requires large memory space. Routing may become impossible when the problem size becomes larger. We use a effective hierarchical method to handle this problem. The other feature of $(\alpha, \beta)^\star$ algorithm is the detour. Detour connects two given pins on a larger routing area than the minimum bounding box including the two pins. The detour may result in more chances of completing routing in the limited routing space. As a result, it may lead to using fewer routing layers.





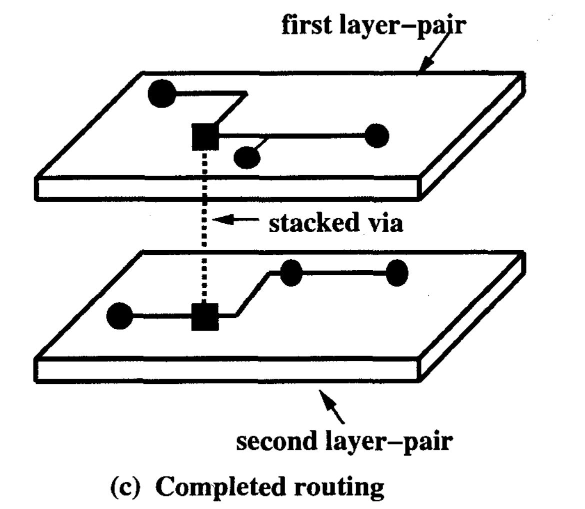


Fig. 2. MLR Algorithm

Fig. 2 can be used to illustrate the basic idea of our MLR algorithm. Assume there is a six-pin net to be connected, as shown in Fig. 2(a). Node 0 is source of the net and all the others are sinks. As shown in Fig. 2(b), SOAR algorithm first builds the initial Steiner subtree on the first layer-pair by connecting the source with sink 5 which is farest from the source. Then sink 4 is connected to the subtree using Steiner node p_{st}, while minimizing the Elmore delay at sink 5. During this process, $(\alpha, \beta)^*$ algorithm is used twice to connect node pair (0, 5) and (4, p_{st}) respectively, using the shortest path and the fewest vias. When the partial routing is completed, the limited space on the first layer-pair may not allowing connecting the remaining sinks to the subtree. In this case, layer assignment algorithm moves all the unrouted sinks into the second layer-pair and chooses a stacked via to minimize the time delay at sink 5, as shown in Fig. 2(b). Then a similar routing is carried out on the second layer-piar to complete the whole routing, which is shown in Fig. 2(c). We shall describe each algorithm above in detail in the remaining sections.

3.2 Layer Assignment Algorithm

Given signal net set $\aleph$ consisting of 2-pin and/or multi-pin nets. Each net $N \in \aleph$ is to be connected by a routing tree T_N. N contains a source pin n_0, with the remaining pin sinks. Let K be the number of routing layer-pairs, $\aleph_{routed}$ record the nets that have been completely routed, and $T_{N,K}$ be the routing subtree of net N in Kth layer-pair. The layer assignment (LA) algorithm is shown in Fig. 3. During the process of layer assignment, SOAR completes either the routing of an entire net or only parts of the net which forms a subtree t on current routing layer-pair. When current layer-pair routing ends, SOAR makes a list for the uncompleted nets and moves them into next routing layer-pair. Therefore, we use a series of subsets $T_{N,K}$ to record these subtrees of net N on different layers.

```
LA Algorithm
Input: Signal net set $\aleph$
Output: $T = \{T_N = \{T_{N,k}, k \leq K\}, \forall N \in \aleph\}$

1.  $K = 0, \aleph_{routed} = \Phi, T_N = \Phi, \forall N \in \aleph$
2.  while $|\aleph_{routed}| < |\aleph|$ do
3.    $K = K + 1$
4.    $T_{N,K} = \Phi, \forall N \in \aleph$
5.    for $\forall N, N \notin \aleph_{routed}, N \in \aleph$ do
6.      $t = SOAR(N, T_N)$
7.      $T_N \leftarrow T_N \cup t$, $T_{N,K} \leftarrow T_{N,K} \cup t$
8.      if $T_N$ covers $N$ then
9.        $\aleph_{routed} \leftarrow \aleph_{routed} \cup N$
10. output routing tree set $T$
```

Fig. 3. Layer Assignment Algorithm

3.3 SOAR Algorithm

SOAR (Steiner Optimal Area Routing) routes a multi-pin net $N = \{n_0, n_1, ..., n_n\}$ in the Manhattan plane. Let $T = (V, E, V_s, B)$ be a routing tree. A node $n_i \in V$ is a pin of net N which is connected to the routing tree T. Each edge $e_{ij} \in E$ connects two nodes n_i and n_j using h-segments and/or v-segments. V_s and B are the Steiner node set and bend set, respectively, in routing tree T. V_s and B are iteratively generated by the $(\alpha, \beta)^*$ Algorithm which, described in the following subsection, connects the node $n_i \notin N$ to the routed subtree of net N.

Let d_i denote the time delay at sink $n_i \in N$. SOAR algorithm is described in Fig. 4. The input of SOAR algorithm is a subtree $T_N, N \in \aleph$. T_N may be an empty set if no pin of net N has been routed in all the current layer-pairs. In step 2 of Fig. 4, we select $n_c \in N$ that is farest to the source n_0. Then we minimize the time delay at sink n_c during the construction of routing tree of net N. Note that the main loop from line 7 to 16 in Fig. 4 may terminate without completing routing of the net N when any two pins of N can not be connected on current layer-pair. In this case, only subtree of net N is built, and the remaining pins are to be routed on next layer-pairs. Procedure alphaBetaStar() carries $(\alpha, \beta)^\star$ algorithm. Obviously, by simply setting $T_N = \Phi$ and removing line 4 and 12, SOAR will become a general Steiner optimal area routing algorithm. Compared with all existing multilayer routing algorithm, SOAR algorithm completes the routing using fewer vias, which is shown by our experiments in section 4.

```
SOAR Algorithm
Input: Signal net set $N$, subtree $T_N$
Output: Steiner subtree $T$ extending $T_N$

1.  $T = \Phi$
2.  find $n_0 \in T_N, n_c \in N, n_c \notin T_N$
3.  $flag = alphaBetaStar(n_0, n_c, B)$
4.  if flag is false goto 17
5.  $V = \{n_0, n_c\}$, $E = (n_0, n_c)$, $V_s = \Phi$
6.  $T = (V, E, V_s, B)$
7.  while $|T_N| < |N|$ do
8.      find $n_i \notin V$, $n_i \in N, n_i \notin T_N$
9.          and $\{n_j, n_k\} \in V$ which minimizes $d_c$
10.     find $v_s$ on $e_{jk}$ nearest to $n_i$
11.     $flag = alphaBetaStar(n_i, v_s, B')$
12.     if flag is false goto 17
13.     $E \leftarrow E \cup \{(n_i, v_s)\}$
14.     $V \leftarrow V \cup \{n_i\}$, $B \leftarrow B \cup B'$
15.     If $v_s \notin V$ then $V_s \leftarrow V_s \cup \{v_s\}$
16.     if $v_s \in B$ then $B = B - v_s$
17. output $T = (V, E, V_s, B)$
```

Fig. 4. SOAR Algorithm

3.4 $(\alpha, \beta)^\star$ Algorithm

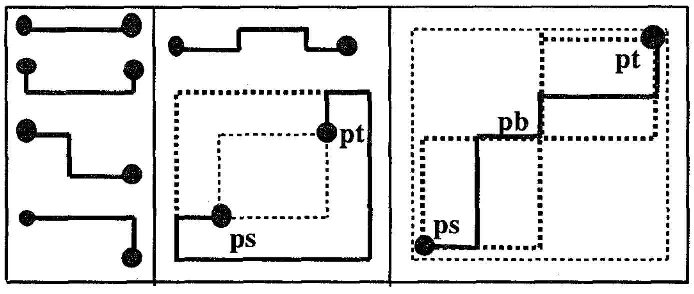
$(\alpha, \beta)^\star$ algorithm shown in Fig. 5 connects two given nodes p_s, the source, and p_t, the target, on the Manhattan plane. Parameters α, β are used to help measure the routing cost. A cost penalty may be imposed on the longer wirelength and the vias (or bends) used. This algorithm extends the existing (α, β) routing algorithm in [8] in two aspects. One is the hierarchical mechanism which enables us to handle larger routing problems by reasonably using the limited memory space. The other is to restrict the route within a minimum bounding box containing the pins p_s and p_t. This helps decrease the memory space requirement and speeds up the algorithm. On the other hand, the strategy also limits the routing ability. Therefore, we use a detour strategy to enhance the routing ability of $(\alpha, \beta)^\star$ algorithm.

Fig. 6 shows the examples of $(\alpha, \beta)^\star$ routing. Some cases of 0-Via routing, 1-Via routing and 2-Via routing are illustrated in the left of Fig. 6. They use some heuristics to connect two nodes using fewest vias and shortest wirelength if possible. The middle of Fig. 6 shows how to detour. A simple detour is applied when two nodes lying on a straight line can not be connected by a single segment. It uses four vias and little longer wire to complete the routing. In other cases, detour may enlarge the minimum bounding box including p_s and p_t, to obtain more chance to complete the routing. In the middle of Fig. 6, the minimum bounding box is indicated in the thin dotted lines, and the enlarged bounding box is indicated in the thick dotted lines. Detour completes the routing with longer wirelength and more vias within the enlarged bounding box. Obviously, the detour must be restricted to avoid too much increase in total wirelength and vias. We allow about 10% of total wirelength in the detour process. Experiments shows that the detour is very useful in decreasing the total number of routing layers. The basic idea of hierarchical (α, β) routing is shown in the right of Fig. 6. When the larger bounding box indicated in the thin dotted lines needs too much memory space and seriously speeds down the routing, the bounding box is divided into two smaller boxes indicated in the thick dotted lines by choosing a node p_b near the center of the box and not being occupied by any other routed net. Then (α, β) algorithm is used to route these smaller boxes one by one. Finally, a complete path from p_s to p_t is obtained. Of course, the two boxes may be further divided into four or more snaller boxes if needed.

```
$(\alpha, \beta)^\star$ Algorithm
Input: nodes $p_s, p_t$, parameters $\alpha, \beta$
Output: A route connecting $p_s$ and $p_t$

1. 0-Via routing
2. 1-Via routing if 0-Via routing fails
3. 2-Via routing if 1-Via routing fails
4. hierarchical $(\alpha, \beta)$ routing if 2-Via routing fails
5. detour if $(\alpha, \beta)$ routing fails
6. output the route from $p_s$ to $p_t$
```

Fig. 5. $(\alpha, \beta)^\star$ Algorithm



0–2 via routing **detour** **hierarchical routing**

Fig. 6. Examples of $(\alpha, \beta)^\star$ Routing

4 Experiments

The MLR multilayer routing algorithm has been implemented in the C programming language, and all experiments are performed on a DEC Station 5000/125.

Seven benchmarks, including two ICs, ami49T and xeroxT, and five MCMs, spert, MCC1PG, MCC1, MCC2-75 and MCC2-45, are used for the experiments. The benchmarks ami49T and xeroxT are obtained by adding appropriate timing information to the CBL/NCSU benchmarks ami49 and xerox respectively. Spert is an MCM developed at the International Computer Science Institute in Berkeley, California. All other MCM benchmarks are industrial routing examples provided by MCC. Benchmark MCC1PG is same with MCC1 except it has more three power and ground nets. MCC2-75 and MCC2-45 are generated from same MCC benchmark by using different routing pitch, 75 μm and 45 μm, respectively. Table 1 lists the main characteristics of the test examples, where $Pin_{max}(net)$ is the maximum number of pins connected by one single net, and S_Size is the substrate size.

For all tests, we evaluate the routing quality using the number of layers, the number of vias, total wirelength, maximum sink delay and the running time consumption (CPU). The experimental results of MLR are shown in Table 2. As shown, MLR algorithm can handle both IC and MCM multilayer routing problems whose size can vary from small one (e.g., xeroxT) to very large one (e.g., MCC2-45). For all the cases, the reasonable maximum sink delays are obtained, which illustrates the timing-driven multilayer routing is successful. Moreover, MLR routes all the benchmarks using very few vias, short wirelength, reasonable number of routing layers, and limited running time (CPU). In case of benchmark MCC1PG, the routing of power and ground nets are completed on a particular routing layer-pair, and then, the limited residual space in this layer-pair is used to route as many signal nets as possible. This strategy help MLR complete the whole routing of MCC1PG using only four routing layers which is same with the case of benchmark MCC1 where the power and ground nets are not considered. These results show that MLR algorithm is very successful for all the benchmarks tested.

Results have previously been reported for MCG [1], SLICE, V4R and 3-D Maze in [4] for all the MCC benchmarks. The results of comparing MLR with them are shown in Table 3 and 4. As shown, MLR uses fewest vias and shortest wirelength for all cases. For example, comparing with MCG, MLR decreased the vias used by 56.8% in case of MCC1 and 44.5% in case of MCC2-75. For the most difficult benchmark MCC2-45 where MCG, SLICE and Maze do not work, MLR completes the routing with fewer vias, shorter wirelength and much less running time than V4R. Furthermore, in most of the cases tested, MLR runs faster than all other routers listed in Table 3 and 4. In the case of MCC2-75, MLR completed routing with two more routing layers than MCG, but MLR's high routing quality including fewer vias, shorter wirelength and less running time makes it worth the increase.

5 Conclusions

A new timing-driven multilayer routing algorithm, MLR, for both IC and MCM designs is proposed in this paper. MLR algorithm was tested on several industrial benchmarks. MLR was shown to have three main features. First, the algorithm first exactly obtains the time delay of all benchmarks tested based on Elmore delay model for the multilayer detailed routing designs. Second, comparing with the existing approaches, MLR algorithm uses fewest vias for all the benchmarks tested and the improvement is very significant. Lastly, some signal nets are routed in the limited routing space on the power and ground layer-pair, which is very useful in decreasing the number of layers needed to complete the routing.

Acknowledgements

The authors wish to acknowledge the support of SRC and NSF under the grants 95-DC-324 and MIP-9117328.

References

[1] J.D. Carothers, D. Li, "A Multilayer MCM Autorouter Based on the Correct-by-Design Approach", Proc. 8th Annual IEEE Intl. ASIC Conf. and Exhibit, pp.139-142, Sept. 1995.

[2] J.D. Cho, K.F. Liao, S. Raje, M. Sarrafzadeh, "M^2R: Multilayer Routing Algorithm for High-Performance MCMs", IEEE Trans. on CAS-I, Vol.41, N0.4, pp.253-265, 1994.

[3] J.M. Ho, M. Sarrafzadeh, G. Vijayan, C.K. Wong, "Layer Assignment for Multichip Modules", IEEE Trans. on CAD, Vol. 9, No. 12, pp. 1272-1277, 1990.

[4] K.Y. Khoo, J. Cong, "An Efficient Multilayer MCM Router Based on Four-Via Routing", 30th DAC, pp. 590-595, 1993.

[5] K.Y. Khoo, J. Cong, "A Fast Multilayer General Area Router for MCM Design", IEEE Trans. on CAS-II, Vol. 39, No. 11, Nov. 1992.

[6] D. Staepelaere, J. Jue, T. Dayan, W.W.W. Dai, "Surf: A Rubber-Band Routing System for Multichip Modules", IEEE Design and Test of Computers, Vol.10, pp.18-26, Dec. 1993.

[7] Q. Yu, S. Badida, N. Sherwani, "Algorithmic Aspects of Three Dimensional MCM Routing", 31st DAC, pp. 397-401, 1994.

[8] T.C. Hu, M.T. Shing, "The $\alpha - \beta$ Routing", in VLSI Circuit Layout: Theory and Design, Ed. T.C. Hu and Ernest S. Kuh, New York, 1985, IEEE Press, pp. 139-143.

[9] H. Esbensen, E.S. Kuh, "An MCM/IC Timing-Driven Placement Algorithm Featuring Explicit Design Space Exploration", MCMC'96, pp. 170-175, 1996.

[10] D.S. Wang, Ernest Kuh, "Performance-Driven Interconnect Global Routing", Proc. 6th Great Lakes Symp. on VLSI, pp. 132-136, 1996

[11] D.S. Wang, E.S. Kuh, "A New Performance-Driven Multilayer IC/MCM Routing Algorithm", submitted to DAC'97.

[12] T. Xue, E.S. Kuh, D.S. Wang, "Post Global Routing Crosstalk Risk Estimation and Reduction", to appear in proc. ICCAD'96.

Table 1. Benchmark Specifications

Parameters	IC		MCM				
	xeroxT	ami49T	spert	MCC1PG	MCC1	MCC2-75	MCC2-45
# of Nets	203	408	248	802	799	7118	7118
# of Pins	698	953	1168	2496	2496	14659	14659
$Pin_{max}(net)$	72	24	190	7	7	4	4
Grid Size	1593x1128	1831x1387	2138x2119	599x599	599x599	2032x2032	3386x3386
S_Size (mm^2)	6.37x4.51	7.32x5.55	85.52x84.76	45x45	45x45	1523.4x152.4	152.4x152.4

Table 2. Exprimental Results of MLR

Parameters	IC		MCM				
	xeroxT	ami49T	spert	MCC1PG	MCC1	MCC2-75	MCC2-45
# of Layers	4	4	4	4	4	6	4
# of Vias	798	1073	1576	3010	2481	19041	21665
Wirelength	240870	416821	539900	413732	370886	5434400	9050509
Time Delay (ns)	0.671	0.565	12.582	0.189	0.186	0.144	0.194
CPU (sec.)	5	9	13	1281	376	3079	2967

Table 3. Comparisons of number of layers and vias

Tests	# of layers					# of vias				
	MLR	MCG	V4R	SLICE	Maze	MLR	MCG	V4R	SLICE	Maze
MCC1	4	4	4	5	5	2481	5747	6993	6386	8794
MCC2-75	6	4	6	7	-	19041	34311	36438	47864	-
MCC2-45	4	-	4	-	-	21665	-	36473	-	-

Table 4. Comparisons of wirelength and running time

Tests	Total wirelength					CPU (sec.)				
	MLR	MCG	V4R	SLICE	Maze	MLR	MCG	V4R	SLICE	Maze
MCC1	370896	378707	394272	402258	397221	376	540	180	720	6000
MCC2-75	5434400	5695039	5559479	5902818	-	3079	12720	3606	29700	-
MCC2-45	9050509	-	9130705	-	-	2967	-	5520	-	-