The MCG Autorouter for Multichip Modules

Jo Dale Carothers and Donghui Li

***Abstract*— A multilayer, multichip module (MCM) router, called MCG, is introduced for x-y routing. An efficient method has been derived to allow candidate routes for the nets to be considered simultaneously for compatibility, rather than incrementally extending routes or routing one net at a time, as in many other techniques, and thus allowing incorporation of accurate models for determining the potential for crosstalk and delay problems during the routing process. In comparisons with other routers on industrial benchmarks, the MCG router has shown substantial improvement in routing density, number of layers, number of vias, and total interconnect length over routers such as V4R and SLICE. Our test results show up to 18% improvement in via count and up to 33% improvement in the required number of routing layers for these examples over V4R. One of the benchmarks presented contains 37 VHSIC gate arrays, over 7000 nets, and over 14 000 pins (pads).**

***Index Terms*—CAD tools, general area routing, multichip modules, physical design.**

I. Introduction

LIMITATIONS of packaging and interconnection technology have led to the inability to take full advantage of the advances in semiconductor fabrication technology. Multichip modules (MCM's) allow IC's (dies) to be placed on a common routing substrate and then to be incorporated into a single package, thus offering a packaging technology that allows for high-performance design. This results in increased system speed by lowering transmission delay between chips, decreased system size, and the possibility of decreased system power requirements. However, multichip modules also introduce additional design constraints in terms of electrical and thermal characteristics which must be considered in the design process. In this paper, we address the problem of MCM and high-density printed circuit board (PCB) routing. The MCM routing problem is also much more complex than its IC or PCB equivalents since the full routing area (minus obstacles such as thermal and signal vias) is available for signal distribution. It becomes a full three-dimensional (3-D) problem as well, since the number of layers can be quite large. In addition, interconnects must be treated as lossy transmission lines, which means that measures must be taken to minimize the effects such as crosstalk.

Several algorithms have been proposed for MCM routing [2]–[5], [11]–[13], [16]. Three-dimensional maze routing is a common technique [9]. However, it suffers from large memory requirements since the entire three dimensional grid must be stored, and it is very sensitive to the ordering of the nets. In addition, it tends to result in the use of a large number of vias and does not lend itself to global optimization of routes, since nets are routed independently. Khoo and Cong have proposed two general area routers, SLICE [11] and V4R [12], [13], based on growing routes in a column-by-column approach. SLICE is a planar routing algorithm that operates on a layer-by-layer basis. As the routes are constructed column by column, bends (vias) will be added when obstacles such as pins or other vias are encountered. SLICE produces improved routes over the 3-D maze router. However, in order to complete the routing, a maze router is used to route nets that were not completed during the planar routing phase, thus slowing the algorithm and introducing additional vias. They showed that V4R can produce satisfactory routing solutions that require no more than four vias per route, reduced total wire length, and reduced computation time, relative to SLICE. Sarrafzadeh *et al*., have introduced $\mathbf{M^2R}$ [5], as well as an earlier algorithm [2], [4], [16] for pin redistribution and routing. The $\mathbf{M^2R}$ algorithm is based on a net-by-net approach that combines single-layer routing for the most critical nets followed by x-y routing for less critical nets. Only results on MCM's with less than 1200 nets were reported, so it cannot be stated how well it performs on denser circuits. In addition, it still routes one net at a time, thus limiting global optimization. They have incorporated a simple crosstalk model that attempts to prevent two wires from being routed too close to each other for more than a predetermined wire length. In related work, techniques such as hierarchical routing and rubber band routing have been used successfully for some MCM technologies, such as silicon-on-silicon [6], [7], [15], [19]. The MCG multilayer MCM router developed in the work presented here performs x-y routing. An efficient method has been derived to allow candidate routes to be considered simultaneously for compatibility, rather than incrementally extending routes or routing one net at a time, as in many other techniques. Therefore, it is possible to incorporate accurate models for determining the potential for crosstalk and delay problems during the routing process [8].

The MCG router has been tested on several examples. These include the industrial benchmarks from MCC. In comparisons with other routers, the MCG router has shown substantial improvement in number of vias, in routing density, number of layers (where possible), and total interconnect length over routers such as V4R and SLICE on industrial benchmarks. One of the benchmarks presented contains 37 VHSIC gate arrays, over 7000 nets, and over 14 000 pins (pads). In addition to the improved geometrical results, it provides the flexibility for integrating electrical characteristics into the routing process,

Manuscript received April 1995, revised October 1998. This paper was recommended by Associate Editor J. Nossek.

J. D. Carothers is are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721 USA.

D. Li is with QuickLogic, Sunnyvale, CA 94089 USA.

Publisher Item Identifier S 1057-7122(99)03888-X.

rather than solely relying on postdesign electrical analysis followed by rip up and reroute.

II. Problem Formulation

An instance of the problem is defined by a set of chips M, a set of nets N, a set of I/O terminals P, and a multilayer routing substrate consisting of multiple signal distribution layers. It is assumed that the I/O terminals are brought to the first signal distribution layers using distribution vias. It is also assumed, as in other work, that a Manhattan grid is superimposed on each routing layer. The separation of grid points is determined by the pitch. Vias are used to connect signal wires on different wiring layers. Stacked vias (two or more vias stacked vertically on top of each other) are used used to connect wires in nonadjacent layers. Interconnection vias are used to connect wires in adjacent layers.

The goal of this work was to develop an efficient routing algorithm that takes a more global approach, based on the consideration of the compatibility of a set of candidate routes. This algorithm allows the routing of multiple nets simultaneously, while limiting the memory and computational requirements. This will allow a more global solution, as well as allow the incorporation of more accurate crosstalk models. In addition, the algorithm guarantees that no route will require more than five vias (actually almost all routes are limited to no more than four vias and it is a simple extension to limit routes to four vias if desired) which is important for higher level delay estimation as discussed in [13]. Also, most routes will be completed with wire length equal to the Manhattan distance between terminals, thereby reducing total wire length.

The overall quality of routing solutions can be measured in terms of total wire length, number of vias, number of layers, and routing density. Delay constraints are affected by total wire length. In addition, the shorter the wire length, the less space that routes use, thereby allowing more routes per layer. Since vias represent discontinuities, the goal is to minimize the total number of vias. Costs are greatly affected by the number of layers required to route the design. Therefore, minimizing the total number of layers is a priority. In addition, the more effectively that the routing algorithm can use the area in each layer, the higher the routing density. This will also translate to reduction in the number of layers.

III. The Algorithm

For x-y routing, two adjacent layers are routed at a time, with all horizontal wires on one layer and all vertical lines on the other layer. Stacked vias and interconnection vias connect adjacent wires on different layers. Once no further routing can be completed on a given pair of layers, then the next pair is routed. This process continues until all nets have been routed and guarantees that all nets will be routed. Priority is placed on creating routes with two or fewer vias to reduce the total number of vias. For each pair of layers, the algorithm consists of three main phases. The primary difference between the phases is the method used for constructing the candidate routes for each net.

We first discuss the technique used for handling multiterminal nets and then describe the three phases of the routing algorithm.

A. Multiterminal Nets

Although most nets are two terminal nets, it is also necessary to handle multiterminal nets. The technique used is based upon a traditional minimal spanning-tree approach combined with techniques for Steiner point insertion. For each multiterminal net, the minimal spanning tree is determined. For a t terminal net, the minimal spanning tree will contain $t-1$ edges. The t terminal net is then decomposed into the corresponding $t-1$ two terminal nets. In the routing algorithm, these routes are then treated as two terminal nets. This effectively allows the insertion of Steiner points for some nets during the routing process and in a postprocessing step, and thus can reduce the total wire length of multiterminal nets below that of the minimal spanning tree. The decomposition of the multiterminal nets occurs during initialization. Therefore, for the purpose of discussing the details of the routing algorithm, it will be assumed that all nets have two terminals.

B. MCG Routing Algorithm

The goal is to achieve the highest possible routing density on any given pair of layers, while minimizing total wire length and the number of interconnection vias. For any given net there will exist many possible ways of routing that net. However, some of these possible routes may interfere with each other in terms of geometrical and/or electrical (i.e., crosstalk) constraints. Ideally, the goal would be to find the largest subset of these routes from the set of all possible routes that are compatible with each other. The MCG algorithm is motivated by this ideal and attempts to approximate this goal by considering a small number of possible routes for each net, and then constructing what we term a compatibility graph. Then this compatibility graph is reduced in polynomial time to yield a subset of routes which are fully compatible. This approach balances the need for a global approach which will improve routing solutions with the requirement for efficient computation speed and memory requirements. It is shown that by using a small number of candidates for each net, excellent routing results can be obtained in reasonable time.

It is first necessary to define a few terms. The bounding box for a net is defined to be the smallest rectilinear box containing all terminals in the net. Whenever we state that a route is constructed within the bounding box, we are limiting those routes to be equal to the Manhattan distance between source and target terminals. A candidate route is a specific route under consideration for a given net. We also define a route compatibility graph (RCG). An RCG is a graph $G = (V, E)$ where V is a set of vertices representing candidate routes and E is a set of edges such that edge e_{ij} exists if and only if routes v_i and v_j cannot be routed on the same x-y, pair of layers due to either geometrical or other defined constraints (i.e., crosstalk). Note that no edge will exist between candidate routes for the same net, since at most one would be implemented. Thus, G is an n-partite graph. Two vertices in the graph are considered

Let ϵ be 0.5–1.0% of the total number of nets

Repeat Phase 1
  Find c candidate routes per net within bounding box (0–2 vias)
  Construct compatibility graph
  Reduce graph until no incompatibilities remain
  Select routes
Until number of additional nets routed $< \epsilon$
Repeat Phase 2
  Find c candidate routes per net within bounding box (3–5 vias)
  Construct compatibility graph
  Reduce graph until no incompatibilities remain
  Select routes
Until number of additional nets routed $< \epsilon$
Repeat Phase 3
  Find c candidate routes per net extending bounding box (4 vias)
  Construct compatibility graph
  Reduce graph until no incompatibilities remain
  Select routes
Until number of additional nets routed $< \epsilon$

Fig. 1. Overview of MCG routing algorithm.

compatible if no edge is adjacent to both vertices. As a result, the corresponding routes can be compatibly routed on the same x-y layer pairs. Results of experimental testing show that it is only necessary to consider a small number of candidates for each net to obtain an excellent routing solution. As the size of the problem increases, it was found that selecting a smaller number of candidates and running each phase more iterations was much faster and still resulted in high quality solutions. (5–20 candidates routes were used for MCC1 and 2–8 candidate routes were used for MCC2, as shown in the next section. Extensive testing indicated that a large number of candidate routes increased the computation time and did not significantly improve the results.) Therefore, the computation and memory requirements are very reasonable. An overview of the algorithm is shown in Fig. 1. This is followed by a detailed description of the algorithm.

1) Phase 1: Let c be the predetermined number of candidate routes to be constructed for each net. Candidate routes are constructed as follows. If not prevented by obstacles (vias, nets routed in prior iterations of Phase 1, etc.), then one via routes are constructed along the bounding box perimeter. At most two such routes exist. Routes with zero vias can only occur if the source and target terminals are in the same row or same column.

If c candidate routes have not been constructed, then the two via routes are constructed. An example of a two-via route is shown in Fig. 2. In order to describe the algorithm for constructing these routes, several terms must be defined. The upper, left-hand point in the routing grid is labeled $(0, 0)$. Given a two-terminal net, the terminal with the smallest column coordinate is called the source terminal and is referenced by row and column coordinates (x_s, y_s). The second terminal is referred to as the target and is referenced by (x_t, y_t). $y_s \max$ is the number of consecutive column grid points currently open in row x_s, starting at y_s in the direction of y_t. $x_s \max$ is the number of consecutive column grid points currently open in column y_s, starting at x_s in the direction of x_t. $x_t \max$ and $y_t \max$ are defined similarly. The two via routes are constructed as shown in Figs. 3 and 4.

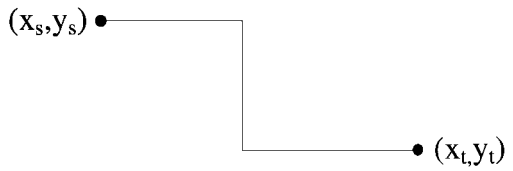

Fig. 2. Example route with two vias.

From these, up to c routes are then selected for each net and are called the candidate routes. The route compatibility graph is then constructed. The next step is to select the largest subset of compatible vertices. This is an NP-complete problem so a heuristic technique is used to select the desired subset. First, the vertex with maximum degree is removed from the graph. This process is repeated until no edges remain. All remaining vertices (routes) are compatible.

To select the actual routes, for each net select one of the remaining candidate routes in the RCG. Note that for some nets no candidate will remain, and for others multiple candidates will remain. If there is no compatible candidate for a given route, then that net must be routed in another iteration. If multiple candidates remain, select only one. Selection can be arbitrary, or can be decided by priority, number of vias, etc. Note that all selected routes will become obstacles in future iterations and routing phases for this layer pair.

2) Phase 2: If not prevented by obstacles (vias, nets routed in Phase 1 or in prior iterations of Phase 2, etc.), then three via routes can be constructed within the bounding box. An example three-via route is shown in Fig. 5. If at least c candidate routes are found, then no more routes are constructed. Otherwise, four or five via routes are constructed within the bounding box. The first c routes constructed for each net become the candidate routes, and all routes constructed during Phase 2 have minimum length, as in Phase 1. Three via routes are constructed by selecting either a point between $(x_s, y_s + 1)$ and $(x_s, y_s + y_s \max)$ or between $(x_s + 1, y_s)$ and $(x_s + x_s \max, y_s)$. Then the router attempts to construct a two-via route between that point and (x_t, y_t), as shown in Fig. 6. Four via routes are constructed by selecting a pair of points, with the first point selected as in the three via case and the second point between either $(x_t - x_t \max, y_t)$ and $(x_t - 1)$ or between $(x_t, y_t - y_t \max)$ and $(x_t, y_t - 1)$. Then the router attempts to construct a two via route between these two points, as shown in Fig. 7. If c candidate routes have not yet been found, additional routes are constructed that have either four or five vias. These routes are constructed by selecting a point (x_p, y_p) inside the bounding box. Then two via routes from (x_s, y_s) to (x_p, y_p) and from (x_p, y_p) to (x_t, y_t) are constructed, as shown in Fig. 8. The algorithm is presented in Fig. 9. The remaining steps in Phase 2 are the same as those in Phase 1.

3) Phase 3: The goal of Phase 3 to is allow routes to extend outside the bounding box, rather than to explore routes requiring more vias within the bounding box. The motivation for constructing routes outside the bounding box is that since the previous two phases explored and routed only within this area, it is likely to be fairly dense already. Therefore, further routes constructed within the bounding box will likely

```
Begin two_via((x_s, y_s),(x_t, y_t))
For each net
    If x_s < x_t
        If y_s + y_smax > y_t − y_tmax
            For columns = (y_t − y_tmax) to (y_s + y_smax)
                If the grid points in rows x_s to x_t are unoccupied
                    Then the route [(x_s, y_s),(x_s,column),(x_t,column), (x_t, y_t)] is feasible
        If x_s + x_smax > x_t − x_tmax
            For rows = (x_t − x_tmax) to (x_s + x_smax)
                If the grid points in columns y_s to y_t are unoccupied
                    Then the route [(x_s, y_s),(row,y_s),(row,y_t), (x_t, y_t)] is feasible
    Else
        This case is the mirror image of the above case.
End two_via
```

Fig. 3. Construction of two via routes.

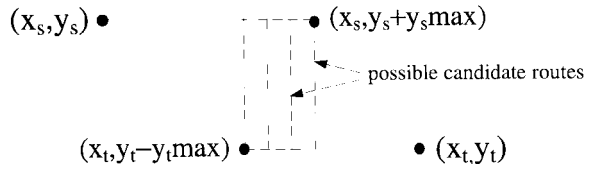

Fig. 4. Possible candidate routes with two vias.

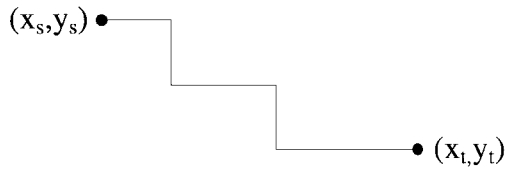

Fig. 5. Example route with four vias.

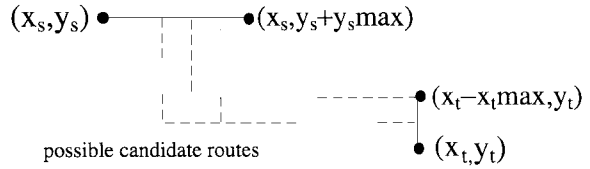

Fig. 6. Example candidate routes with three vias.

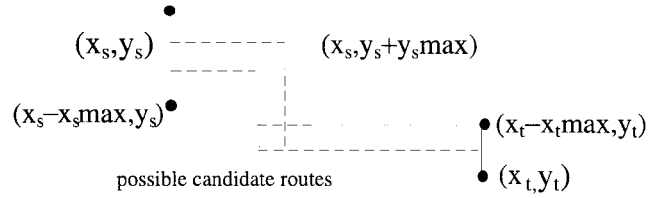

Fig. 7. Example candidate routes with four vias.

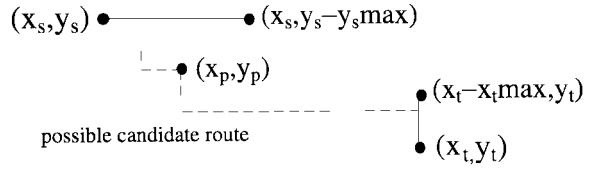

Fig. 8. Example candidate routes with five vias.

require more bends (vias) and longer wire length. By looking for candidates in a slightly enlarged area, the possibility for constructing routes with minimal vias and acceptable wire

```
While (num_found < c)
    /* construct three via routes */
    For row = x_s + 1 to x_s + x_smax
        two_via((row,y_s),(x_t, y_t))
        If ++num_found = c break;
    For col = y_s + 1 to y_s + y_smax
        two_via((x_s,col),(x_t,y_t))
        If ++num_found = c break;
    /* construct four via routes */
    For source_row = x_s + 1 to x_s + x_smax
        For target_row = x_t − x_tmax to x_t − 1
            two_via((source_row,y_s),(target_row,y_t))
            If ++num_found = c break;
    For source_col = y_s + 1 to y_s + y_smax
        For target_col = y_t − y_tmax to y_t − 1
            two_via((x_s,source_col),(x_t,target_col))
            If ++num_found = c break;
While (num_found < c)
    /* construct additional four and five via routes */
    Select next unoccupied grid point (x_p,y_p) within bounding box
        If two_via((x_s,y_s),(x_p,y_p)) &
            two_via((x_p,y_p),(x_t,y_t)) exist
                then ++num_found
```

Fig. 9. Construction of three, four, and five via routes.

```
While (num_found < c)
    Select points p_s and p_t
        modified_two_via((p_s),(p_t))
        If found then ++num_found
modified_two_via
        returns first route constructed by two_via
```

Fig. 10. Construction of routes outside bounding box.

length is more likely. We limit the routes outside the bounding box to four vias, and an optional limitation on the size of the enlarged routing box is allowed. These are the only routes created by this algorithm that are not guaranteed to have minimum length. For (x_s, y_s) the number of adjacent open grid points in all four directions are determined. One of these points, p_s, is randomly selected. Similarly, a point p_t for the target terminal is also selected. Then, the routing algorithm constructs a route between p_s and p_t with at most two vias. This process is repeated until at most c candidate routes are determined for each net, as shown in Fig. 10. These routes will contain at most four vias. Phase 3 then proceeds as in Phase 1.

If only a small number of nets are placed on the final pair of layers, then a postprocessing step may be applied to attempt to route these nets on previous layers. For example,

TABLE I
TEST EXAMPLES

Examples	Pitch micron	# of Chips	# of Nets	# of Pins	Size of Substrate (mm^2)	Grid Size
MCC1	75	6	802	2496	45×45	599 x 599
MCC2-75	75	37	7118	14659	152.4×152.4	2032 x 2032
MCC2-45	45	37	7118	14659	152.4×152.4	3386 x 3386

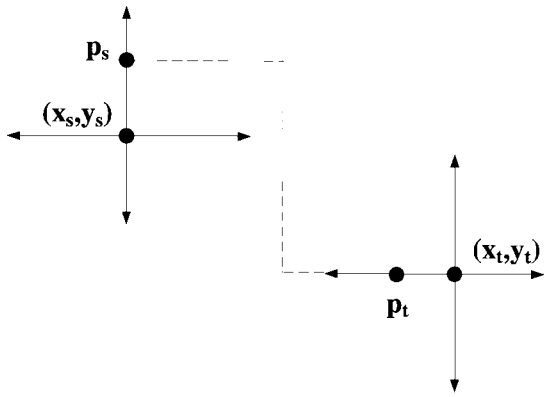

Fig. 11. Routes outside bounding box.

line probing approaches may be used efficiently if the total number of nets is small. By eliminating the final pair of layers, the manufacturing cost can be significantly reduced and is well worth a few additional minutes of processing time.

It is possible to incorporate accurate models for determining the potential for crosstalk and delay problems during the routing process. A crosstalk-avoidance procedure can be incorporated, which uses closed-form estimates of coupled noise along with lookup tables of electrical parameters in order to estimate crosstalk noise quickly during routing [8]. The crosstalk avoidance procedure used in conjunction with MCG represents an improvement over that of IBM's approach [1], since the lookup tables obviate prerouting simulation of the package and because noise waveform timing is considered. For details on implementation and results of incorporating crosstalk avoidance in MCG see [8].

Through use of modified advanced candidate generation techniques and changes in the data structures to allow variable wire widths and multiterminal candidate generation, advanced delay models can be used in conjunction with the MCG router.

The space needed for running the algorithm includes the storage for the nets, the candidate routes, the compatibility graph, and the obstacles. The space for storage of the nets is linear with the total number of nets, and the storage for the candidate routes linear with total number of candidate routes. For the storage of vertices in the compatibility graph, the number of vertices equals the total number of candidate routes, which, again, is linear to the total number of nets. In the worst case, the memory requirement for the edges is $O(n^2)$ where the total number of nets. The memory requirement for the storage of obstacles depends on the forms in which the obstacles are stored. If the obstacles are stored in the form of linked list, then the space for the storage of obstacles is proportional to the total number of obstacles. (Note that line segments of routes can be stored as a single obstacle.) However, computational efficiency is increased by maintaining information about currently open routing spaces.

The time needed for the process of routing includes the time for the construction of candidate routes, the time for building the compatibility graph, and the time for the reduction of the graph. The time complexity for building the compatibility graph is $O(n^2)$ in the worst case. The worst case time for graph reduction is also $O(n^2)$. The time for the construction of the candidate route depends on the type of route and the algorithm for its construction. The worst case complexities for the types of candidates discussed in this paper are discussed below. For type-0 and type-1 routes the time requirement is constant. For type-2 routes, it is linear to $l + w$ where l is the length and w is the width of the bounding box for the net. For the type-3 route, it is proportional to $l \times w$. For the type-4 route, it is proportional to $l \times w \times (l + w)$. For the type-5 route, it is proportional to $l^2 \times w^2$. For type-X, it is similar to the type-4 route only with a larger bounding box. For the type-Z route, it is linear to the number of line segments generated in the line-probe algorithm. Except for the type-Z route, the type-5 route is the most time consuming to construct and it uses the largest number of vias for the interconnection of the net. Therefore, the use of type-5 route is limited. These merely represent examples of possible candidate generation techniques. Many other possibilities are available.

IV. RESULTS

The MCG router was tested on several examples including industrial examples from MCC. The MCC benchmarks represented the largest examples in terms of number of pins and nets. These results are reported here. The characteristics of these circuits are described in Table I. Results have previously been reported for V4R, SLICE, and the 3-D Maze router for these circuits. The results of comparing MCG with these routers are shown in Tables II and III. As shown, the MCG router used fewer layers on MCC2-75, which significantly reduces the manufacturing cost and complexity and used less total wire length than the other routers for the other examples. Table IV shows that MCG achieved high routing densities (over 90% for MCC2-45) on the first pair of routing layers and finished the remaining routing on the next pair. In addition, MCG significantly improved the via count over all the other routers (up to 18% over V4R, 28% over SLICE, and 35% over 3-D Maze).

A lower bound can be determined for the total wire length. This lower bound can be computed as in [13]. The equation

TABLE II
COMPARISON WITH OTHER ROUTERS. — INDICATES A ROUTER WAS UNABLE TO BENCHMARK. *NOTE THAT SINCE MCG USED FEWER LAYERS THAN V4R FOR THIS BENCHMARK IT REQUIRED SLIGHTLY MORE WIRE LENGTH

Examples	Number of Layers				Total Wire Length			
	MCG	V4R	SLICE	3D Maze	MCG	V4R	SLICE	3D Maze
MCC1	4	4	5	5	376478	394272	402258	397221
MCC2-75	4	6	7	—	5690977*	5559479	5902818	—
MCC2-45	4	4	—	—	9129743	9130705	—	—

TABLE III
TOTAL NUMBER OF VIAS

Example	MCG	V4R	SLICE	3D Maze
MCC1	5742	6993	6386	8794
MCC2-75	34300	36438	47864	—
MCC2-45	33283	36473	—	—

TABLE IV
PERCENT COMPLETED ROUTES AFTER x LAYERS.

Examples	1-2	3-4
MCC1	86.00%	100.00%
MCC2-75	72.65%	100.00%
MCC2-45	90.37%	100.00%

TABLE V
LOWER BOUND ON WIRE LENGTH. *NOTE THAT IF THREE LAYER PAIRS WERE USED THE WIRE LENGTH WOULD DECREASE BUT COST WOULD INCREASE

Examples	Lower Bound	MCG	Ratio
MCC1	343767	376478	1.10
MCC2-75	5362181	5690977*	1.06
MCC2-45	8935372	9129743	1.02

for computing this lower bound $b(n)$ for the wire length of net n is

$$b(n) = \max\left\{m(n), \frac{2}{3}\mathrm{MST}(n)\right\} \tag{1}$$

where $m(n)$ is $1/2$ the perimeter of the smallest bounding box containing all terminals of net n and $\mathrm{MST}(n)$ is the length of the minimal spanning tree for net n. The $2/3$ factor is derived from the fact that in Manhattan routing the wire length of the minimum spanning tree is at most 1 $1/2$ times the wire length of the minimum Steiner tree [10]. The lower bound on total wire length B is given by

$$B = \sum_{n \in N} b(n). \tag{2}$$

The comparison of MCG results with this lower bound is shown in Table V.

The MCG routing algorithm was implemented in C++ on a Sun Microsystems SPARCstation 10. The total execution times for the three phases were 0:09, 3:37, and 1:48 (h:min) for MCC1, MCC2-75, and MCC2-45, respectively. On a SPARCstation 2 using C implementations, it was reported that V4R required 1:06 and SLICE required 8:15 for MCC2-75 and V4R required 1:37 for MCC2-45. Although MCG required more computation time than V4R, the improved quality (fewer vias, higher density, fewer layers (where possible), and reduced wire length, etc.) and the flexibility for the future incorporation of other incompatibility constraints (i.e. crosstalk) as edges in the RCG's make it worth the increase. These extensions are the subject of ongoing work.

V. CONCLUSION

The MCG router was presented for multilayer MCM routing. The router was tested on several examples including industrial benchmarks. MCG was shown to produce significantly improved results in terms of number of vias, routing density, total wire length, and number of layers (where possible) on the MCC circuits when compared to routers such as V4R and SLICE while maintaining reasonable computation times. In addition, electrical (i.e., crosstalk) constraints for wire pairs can be easily incorporated during the construction of the RCG's. In ongoing research, additional crosstalk and delay models are being incorporated into the routing algorithm.

In ongoing research enhancements are being added which will significantly increase the speed of MCG while maintaining the routing density. Further extensions will incorporate routability information and consider delay requirements. MCG will also be shown to extend to IC routing.

REFERENCES

[1] H. H. Chen and C. K. Wong, "Wiring and crosstalk avoidance in multichip module design," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1992, pp. 28.6.1–28.6.4.
[2] J.-D. Cho, K.-F. Liao, and M. Sarrafzadeh, "Multilayer routing algorithm for high performance MCM's," *IEEE Design and Test of Computers*, vol. 10, pp. 27–37, Dec. 1993.
[3] J. D. Cho and M. Sarrafzadeh, "Four-bend top-down global routing," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 793–802, Sept. 1998.
[4] J.-D. Cho, M. Sarrafzadeh, M. Sriram, and S.-M. Kang, "High performance MCM routing," *IEEE Design and Test of Computers*, vol. 10, pp. 27–37, Dec. 1993.
[5] J.-D. Cho, K.-F. Liao, S. Raje, and M. Sarrafzadeh, "M^{2}R: Multilayer routing algorithm for high-performance MCM's," *IEEE Trans. Circuits Syst. I*, vol. 41, pp. 253–265, Apr. 1994.
[6] W. W.-M. Dai, "Performance driven layout of thin-film substrates for multichip modules," in *Proc. 1991 IEEE Int. Symp. Circuits Systems*, 1991, pp. 2308–2311.
[7] W. M. Dai, T. Dayan, and D. Staepelaere, "Topological routing in SURF: Generating a rubber-band sketch," in *Proc. ACM/IEEE 28th Design Automation Conf.*, 1991, pp. 41–44.
[8] T. Hameenanttila, J. D. Carothers, and D. Li, "Fast coupled noise estimation for crosstalk avoidance in the MCG multichip module autorouter," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 356–368, Sept. 1996.

[9] A. Hanafusa, Y. Yamashita, and M. Yasuda, "Three-dimensional routing for multilayer ceramic printed circuit boards," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1990, pp. 386–389.
[10] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," *SIAM J. Appl. Math.*, vol. 30, pp. 104–114, 1976.
[11] K.-Y. Khoo and J. Cong, "A fast multilayer general area router for MCM designs," in *Proc. European Design Automation Conf.*, 1992, pp. 292–297.
[12] K.-Y. Khoo and J. Cong, "An efficient multilayer MCM router based on four-via routing," in *Proc. 1993 ACM/IEEE Design Automation Conf.*, 1993, pp. 590–595.
[13] K.-Y. Khoo and J. Cong, "A fast four-via multilayer MCM router," in *Proc. 1993 IEEE Multi-Chip Module Conf.*, 1993, pp. 179–184.
[14] J. J. Licari, *Multichip Module Design, Fabrication, & Testing*. New York: McGraw-Hill, 1995.
[15] B. Preas, M. Pedram, and D. Curry, "Automatic layout of silicon-on-silicon hybrid packages," in *Proc. ACM/IEEE 26th Design Automation Conf.*, 1989, pp. 393–399.
[16] M. Sarrafzadeh, K.-F. Liao, and C. K. Wong, "Single-layer global routing," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 38–47, Jan. 1994.
[17] N. Sherwani, *Algorithms for VLSI Physical Design Automation*. Norwell, MA: Kluwer, 1993.
[18] M. Sriram and S. M. Kang, "iPROMIS: An interactive performance driven multilayer MCM router," in *Proc. 1993 IEEE Multi-Chip Module Conf.*, 1993, pp. 170–173.
[19] D. Staepelaere, J. Jue, T. Dayan, and W. W.-M. Dai, "Surf: Rubber-band routing system for multichip modules," *IEEE Design and Test of Computers*, vol. 10, pp. 18–26, Dec. 1993.

Jo Dale Carothers received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Texas at Austin, in 1985, 1986, and 1989, respectively.

She is an Associate Professor of Electrical and Computer Engineering at the University of Arizona. Her research interests are in the areas of VLSI physical design automation for multichip modules and IC's, including placement and routing of mixed signal designs.

Dr. Carothers has served on numerous conference organizing and program committees. Currently, she is General Chair of the 1999 Southwest Symposium on Mixed-Signal Design. She is a member of Eta Kappa Nu, Tau Beta Pi, and Phi Kappa Phi.

Donghui Li received the B.S. degree in electrical engineering from Zhengzhou Institute of Technology in 1981 and the M.S. degree in electrical engineering from Wuhan Institute of Technology in 1984. He received the Ph.D. degree in electrical and computer engineering from the University of Arizona, Tucson, in 1995.

In 1995, he joined QuickLogic, Sunnyvale, CA, where his work focuses on physical design. His research interests include partitioning, placement, and routing for single and multichip modules.