

Performance Driven Multi-Layer General Area Routing for PCB/MCM Designs

Jason Cong and Patrick H. Madden
UCLA Computer Science Department
Los Angeles, California, 90095

Abstract

In this paper we present a new global router appropriate for Multichip Module (MCM) and dense Printed Circuit Board (PCB) design, which utilizes a hybrid of the classical rip-up and reroute approach, and the more recent iterative deletion[9] method. The global router addresses performance issues by utilizing recent results in high performance interconnect design, while still effectively minimizing global congestion.

With experiments on the maze-routing component of our global router, we show that the choice of routing cost functions can have a significant impact on final solution quality. The results of a number of previously proposed routers may be improved dramatically by adopting the cost functions we suggest here. We also find little evidence of the “net ordering problem” when our cost functions and routing model are applied. The iterative deletion method is shown to improve global solution quality, particularly when high performance interconnect is required. We evaluate the performance of our global router by comparing the congestion of routes produced by our global router to those of a well known MCM router, V4R [14].

Our global router, MINOTAUR, supports arbitrary numbers of routing layers, differing capacities for each layer, pre-existing congestion and obstacles, and high performance interconnect structures (including those which require variable width interconnect).

1 Introduction

Advances in VLSI fabrication technology have resulted in an increasing interest in interconnection and packaging technologies. For many systems, the delay due to circuit interconnections can dominate the total system delay [1, 5]. Interconnect optimization must be considered to achieve high performance at the system level.

One approach that provides many benefits to packaging level interconnection is the Multichip Module (MCM) technology. This technology increases practical packing densities, and removes a level of interconnection. In MCM designs, a number of bare chips are placed directly onto a substrate, with the substrate itself providing the circuit interconnections. Routing problems for MCM designs have many similarities to routing for dense Printed Circuit Boards (PCBs).

Routing within MCM substrates or PCBs, however, can be an extremely difficult problem with a potentially large

number of interconnect layers. A Fujitsu supercomputer design, for example, has over 50 interconnect layers, each with a large routing capacity [10]. The MCM/PCB routing problem in general may be considered to be an extremely large general area routing problem, and cannot be decomposed into small channel or switchbox routing problems easily. Additionally, high performance designs must consider signal crosstalk (due to coupling capacitance between long parallel lines), and variable width wiring (to address impedance mismatches and wiring termination requirements).

The problem of MCM/PCB routing has been considered by a number of authors. One set of approaches perform global and detail routing in a single step.

An early MCM router was presented in [10], and performs a combination of 2D and 3D routing obtain a solution. Computation times for this method were relatively high.

In [13] the SLICE router was presented; it constructs detailed routes through a mixture of planar routing and two-layer maze routing. While this router was superior to a 3D maze router, it requires relatively high computation times. The V4R router [14] was subsequently presented, and efficiently routes layer-pairs one at a time using routes with no more than four vias each.

The M^2R [4] router addresses performance concerns by routing critical nets on a single layer. The remaining nets are routed using $x - y$ layer pair routing.

In [17], a linear programming approach was presented. Randomized rounding is used to obtain integer solutions. A small number of possible topologies are considered, and routes are restricted to a channel intersection graph.

The MCG [2] router considers a small number of possible routes for each net, and attempts to maximize the routing density on any layer-pair by selecting a set of compatible routes from among the unrouted nets. The results of this router are quite good, requiring fewer routing layers than many of the other MCM routers. Performance concerns are addressed by removing nets which exceed signal crosstalk thresholds, and routing them on other layers.

In [20], the MLR router was presented; it first performs layer assignment of nets, followed by Steiner optimal area routing using a hierarchical (α, β) approach (to find low-cost paths between pins).

The SEGRA [3] MCM global router has been described, and also approaches the routing problem by completing pairs of layers in each pass. A simple greedy heuristic is used to select which nets are to be extended or completed during each pass.

While the methods mentioned above consider global and detail routing in a single step, there have also been approaches which consider the problem in two steps.

In [21], the MCM substrate is decomposed into a set of “towers,” rectangular regions containing portions of the routing surface through all layers. They first distribute routing density across a three dimensional surface, using hierarchical decomposition, then determine locations of nets on

the boundaries of the towers, and finally obtain a routing for each tower using a detail router.

We approach the problem in a manner similar to [21], and consider the global routing problem in this paper. In contrast to these previous approaches, our MINOTAUR global router offers the following features.

- We utilize current high-performance variable-width interconnect optimization results, obtaining structures which address delay and signal integrity requirements [6]. We can in fact consider a *set* of high performance interconnect structures for each signal net, and perform congestion optimization from a global perspective.
- We optimize global congestion by not restricting the layers a route may use; some approaches route a single layer-pair at a time, and cannot obtain routing solutions which span multiple or non-paired layers.
- We combine the freedom and flexibility of maze routing solutions, with the global optimization abilities of the iterative deletion method.

While we consider MCM and PCB routing in this paper, the basic techniques used in our router can be easily applied to multilayer VLSI IC routing problems as well.

The remainder of this paper is organized as follows. In Section 2, we present our problem formulation. We also describe our approach to congestion estimation in this section. In Section 3, we describe the MINOTAUR global router. We detail our methods for applying rip-up and reroute in Section 3.1, the iterative deletion approach in Section 3.2, our support for high-performance interconnect structures in Section 3.3, and the integration of algorithms in Section 3.4. Experimental results are presented in Section 4, and we conclude our paper with Section 5.

2 Problem Formulation

Our routing model is similar to that of [21]; we divide the 3-dimensional routing surface into a set of “tiles,” with each tile consisting of a number of layers. We use “tile-layer” to indicate a single layer within a tile. The routing substrate is modeled as a graph, with tile-layers as nodes, and both vias and the “borders” between tile-layers as edges. Preferred-direction routing is obtained by including (or excluding) border edges. Figure 1 shows an example of our routing model.

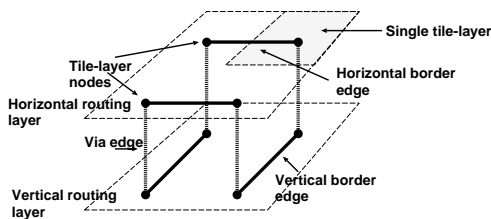


Figure 1: A two-layer four-tile routing problem.

The routing problem consists of determining the interconnect structure for a set of nets N . In addition to basic feasibility concerns (based on physical routing capacity constraints), there are a number of performance issues as well; many of these were detailed in [4]. In this paper, we are interested in meeting delay, signal integrity, crosstalk, and

routing capacity constraints, while minimizing total wire length, maximum congestion, and routing resource costs.

We assume that a detail router will be used to complete the routings of the individual tiles, and determine solution feasibility through the maximum *congestion* of any tile-layer node, border edge, or via edge. We estimate solution congestion as the number of topology edges using a particular routing resource. Note that the border congestion and tile-layer congestion metrics are in general not comparable.

3 The MINOTAUR Global Router

Our approach is to first decompose each non-critical signal net into a set of edges. In most of our experiments, we use a simple minimum spanning tree algorithm, although we can also utilize Steiner tree or high performance interconnect algorithms. Steiner topologies had little impact on solution quality in our experiments. Nets which require high-performance interconnect structures are not decomposed into 2-pin edges, and are handled as described in Section 3.3.

For each edge of the non-critical nets, we wish to find a *path* through the modeled MCM substrate; we model a path as a set of tile-layer nodes, border edges, and via edges.

Our global router contains two basic algorithmic approaches: a traditional rip-up and reroute scheme, and a method based on iterative deletion. We first discuss the details of each, and then show how they are integrated.

3.1 Rip-Up and Reroute

Rip-up and reroute is one of the oldest approaches to the global routing problem; it generally requires long computation times and provides few guarantees as to solution quality. Despite these drawbacks, rip-up and reroute continues to be used in many routing applications. The rip-up and reroute portion of our global router is similar to that of [19], in that we iteratively change our congestion bounds to encourage convergence to a low-congestion solution.

Given the complexity of MCM and PCB designs, we believe that rip-up and reroute may be necessary for aggressive design. We implement rip-up and reroute in our router as follows. Each edge is first routed using a maze router based on Lee’s classic algorithm [15]. We then iteratively rip-up and reroute each edge, continuing for a user-specified number of iterations, or until solution quality stabilizes. Details of our maze routing cost functions, and edge orderings used, are given below.

If high-performance interconnect structures are required, we select a single high-performance interconnect structure with the minimum area for the performance critical net, and do not modify it. The rip-up and reroute process operates only on the edges of non-critical nets, and attempts to find routes “around” the congestion of the critical nets. Congestion optimization of the high-performance nets by the iterative deletion method is more robust, and is detailed below.

3.1.1 Maze Routing Cost Functions

As is done in a number of other routers, we assign a cost to each tile-layer node, border edge, and via edge. Our objective is to assign costs to each resource such that we find a relatively short path (in terms of wiring distance), while avoiding heavily congested areas. We are therefore interested in knowing what cost functions result in the best

performance, a question that has received relatively little attention in the literature.

In [19], the cost of routing in the most congested region is infinite, while the remaining regions have low cost. In [12], the cost of a routing path includes terms for the number of other routes that a path “touches” or “crosses,” as well as two terms to capture the route length in preferred and non-preferred routing directions. In [18], steps in the preferred routing direction were assigned a cost of 1, while non-preferred steps were assigned a cost of 30. In [16], costs were piecewise-linear functions depending on a measure of the utilization and the capacity of each edge. Clearly, there are many possible ways to calculate the cost of using a particular resource.

In general, we expect resources which have high usage or large numbers of obstacles to have relatively high cost. We have explored three cost functions in depth, and describe them here.

We first consider a *step* function, similar to those of [19], [18], and [12], which provides a low cost if congestion falls below a specific threshold value. A high cost results if congestion is at or above the threshold value. Second, we considered a piece-wise linear *slope* function, similar to that of [16], which provides low cost when congestion is below a threshold, high cost if congestion is above *cap*, and a linearly interpolated cost when congestion is between the threshold and *cap* levels. For each iteration of rip-up and reroute, we determine the maximum and minimum congestion for any resource. We modify the *step* and *slope* functions to reflect current congestion levels by adjusting the *threshold* and *cap* parameters. We refer to this as *Adaptive Congestion Estimation (ACE)*, and perform experiments using *ACE-step* and *ACE-slope* cost functions. A third function, which we refer to as a “two-tier” function, is similar to the *slope* function when routing congestion is below a fixed physical capacity. If congestion is above physical capacity, routing costs are increased substantially. These cost functions are illustrated in Figure 2.

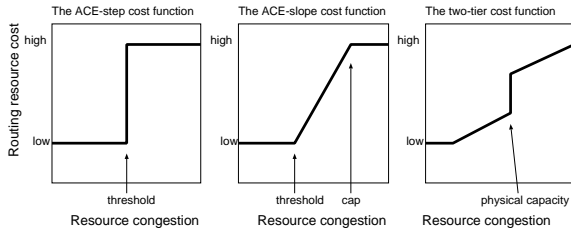


Figure 2: *ACE-step*, *ACE-slope*, and “two-tier” cost functions. These functions are used to determine resource cost during maze routing.

We find that the *ACE-slope* cost function is consistently superior to the *ACE-step* function. Detailed results are given in Section 4, Table 2. Many routing applications which utilize step-like functions ([19], [12],[18], for example) may obtain substantial improvements in results by adopting a slope-like cost function.

Our global router supports *separate* cost functions for each type of resource (tile-layer nodes, border edges, and via edges), and can optimize these independently.

3.1.2 The Net Ordering Problem

A perennial question in global routing problems has been *which net do we route (or reroute) first?* [9, 2]. To evaluate

the importance of net ordering in our implementation, we compared five different edge orderings within the rip-up and reroute framework. Our orderings were low length first, low cost first, the reverse of these, and the default order of our design database.

Surprisingly, there was negligible difference in the results by the various orderings. While examples can be constructed in which certain orderings are clearly superior to others, we found little evidence that this occurs in our experiments. In our model, we have no hard upper limit on routing capacity, and perform rip-up and reroute until solution quality converges. Our study concludes that solution quality measured in our routing model is far more sensitive to the choice of routing cost functions than to the edge ordering.

3.2 The Iterative Deletion Method

Rip-up and reroute is computationally expensive. Through *iterative deletion*, we can obtain high quality results while using rip-up and reroute sparingly. In this section, we describe iterative deletion; Section 3.4 describes integration with rip-up and reroute. The use of iterative deletion was first proposed in [9], and was recently improved in [7].

In many iterative improvement methods, we repeatedly modify a solution, changing a small portion at each step. Rather than representing the problem with a single instance, and changing one aspect of it at each step, the iterative deletion method represents a *set* of possible states. Consider the example in Figure 3, which is a relatively simple planar routing problem with three nets.

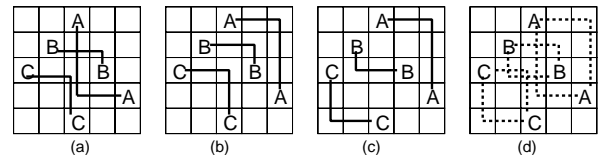


Figure 3: A simple planar routing problem (showing solutions with high congestion, high crosstalk, low crosstalk, and a set of redundant candidate routes).

Figure 3a has high congestion, while Figure 3b has high coupling capacitance (due to parallel wire segments). Figure 3c has both low congestion and low coupling capacitance, and is a *subset* of Figure 3d. The iterative deletion process begins with a redundant set of paths (such as Figure 3d), and iteratively removes a single high-cost redundant path while attempting to minimize congestion and crosstalk among the remaining connections. Interactions between paths (such as coupling capacitance) can also be modeled within this framework.

The set of possible connections for each edge is called the *path set*. In Figure 3d, the path sets for each edge consist of the two single-bend routes. By having several possible routes for each edge available at the start of the iterative deletion process, we can easily determine which paths are high quality. In the example, the lower-left routing for net *C* does not interfere with any other path, allowing the alternative to be removed without degrading solution quality. This approach has a number of significant features. First, we obtain a *global* estimate of congestion, and can determine not only where congestion is likely, but also where it is unavoidable. Second, crosstalk between paths can be modeled and considered during optimization. By selecting a set of routes that are compatible at a *global* level, we can obtain high quality solutions quickly.

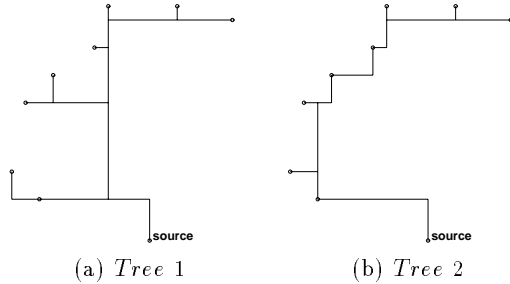


Figure 4: An example of the topologies generated by the *RATS-tree* algorithm for a 9-pin net.

Iterative deletion was shown to be an effective method for selecting a set of channel segments in standard cell IC routing problems [7], and motivates our use of the method here. Our congestion estimation here is similar to the channel density estimation of [7]. We recalculate congestion levels after the removal of each redundant path.

3.3 Performance Optimization

Many approaches for interconnect optimization in high performance design have been proposed. [11] and [5] provide an overview of current interconnect optimization approaches. Recently, *Required-Arrival-Time Steiner trees (RATS-trees)* have been proposed [6] to provide a set of solutions allowing trade-off among signal delay, wave-form, and routing area. This algorithm uses a bottom-up dynamic programming approach, and can synthesize a *set* of high performance structures; Figure 4 shows two possible trees for a 9-pin net.

While in our basic iterative deletion approach considers alternate routings for each topology edge, we consider alternate *trees* for each critical net. Rather than removing a single path between a pair of pins during deletion, we instead remove one of the candidate interconnect trees generated by the *RATS-tree* algorithm. The *RATS-tree* algorithm and our global router support variable-width routing.

3.4 The Hybrid Approach

We have three approaches that form a hybrid between the rip-up and reroute approach, and the iterative deletion method. The first applies iterated rip-up and reroute, and then adds the single bend paths to the path set for each edge. We then perform iterative deletion on the path sets.

Our second approach begins by generating single-bend paths for each edge, and then applies iterative deletion to these path sets. After iterative deletion, we use the maze router to find a new candidate path for each edge, and add it to the path set for the edge. Multiple iterations of this process can produce high quality results quickly.

A third approach applies iterative deletion as above, followed by rip-up and reroute for non-critical nets.

4 Experimental Results

Experiments were performed with the MCC multichip module benchmark circuits *mcc1* and *mcc2*, and also on the three test cases from [14], summarized in Table 1. We report maximum and average congestion levels (C_{max} and C_{avg}) for either the tile-layer metric or the border metric (these two are

Benchmark	No. of Chips	No. of Nets	No. of Pins	Net cardinality			
				2	3	4	5+
test1	4	500	1000	500	0	0	0
test2	9	956	1912	956	0	0	0
test3	9	1254	2508	1254	0	0	0
mcc1	6	802	2495	608	87	50	57
mcc2	37	7118	14659	6699	415	4	0

Table 1: Benchmark circuit parameters. *test1*, *test2*, and *test3* are constructed examples, containing only two-pin nets.

not in general comparable). Average congestion should correlate reasonably well with wire lengths available after completion of detail routing. A complete set of results is available in [8]. We omit experimental results on edge ordering (in which no ordering outperformed another), an exhaustive set of results comparing cost function approaches, and experiments comparing the use of spanning tree and Steiner tree topologies (which also show negligible differences).

Maze routing (which is unoptimized) dominates the run time. MINOTAUR routes *mcc2* in one hour on a Sun Ultra-1, using a single layer 16 by 16 routing graph. Increasing graph resolution increases run times.

4.1 Comparison of Cost Functions

To compare cost function performance, we route each example as a single layer problem, using a 16 by 16 routing grid. We use a low routing cost of 1, and tested combinations with high routing costs of 2, 4, 8, and 16, *ACE* cost function threshold points at 0.2, 0.4, 0.6, and 0.8 of the range between the minimum and maximum congestion level, and also a threshold point at the average congestion level. A summary of these results is shown in Table 2, in which we show the best, worst, and “typical” performance of the rip-up and reroute method for either the step or slope cost functions, and the result of a single pass of iterative deletion considering the rip-up and reroute paths and the single-bend candidate paths. In the “typical” cost function configuration, we use a high routing cost of 16, and the cost function threshold point at the average congestion level.

While the congestion values during the rip-up and reroute process converged for all experiments, they converged to vastly different values depending on the cost function, and the *high*, *low*, and *threshold* values used to configure the cost function; maximum congestion levels differed by as much as a factor of 2.29. We find that the results using the slope cost function are superior to the step based cost function. When the *high*, *low*, and *threshold* values for tile costs are not well suited to the routing problem, a single pass of iterative deletion can improve the results considerably.

4.2 Comparison with V4R

In Table 3, we compare maximum tile-layer congestion values of our approach with the results of the MCM global router V4R [14]. Congestion estimation favors the V4R router, as the MINOTAUR result considers distinct *tree edges*, while the V4R result reported considers distinct nets. For the single layer comparisons, we merge layer assignments of the V4R routing into a single layer. Our global router obtains congestion reductions ranging from 35% to 48% for the single layer model, and reductions of 14% to 38% for the multi-layer model.

Benchmark	Best RR		Best RR + ID		Worst RR		Worst RR + ID		Typical RR		Typical RR + ID	
	C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}
test1 <i>slope</i>	27	23.7	27	23.7	28	23.7	28	23.7	27	24.3	28	24.0
test1 <i>step</i>	33	24.7	30	24.1	58	23.7	33	23.7	33	24.8	30	23.1
test2 <i>slope</i>	49	46.3	51	46.1	51	46.5	51	46.2	50	47.0	52	46.5
test2 <i>step</i>	57	49.1	54	47.7	112	45.9	59	45.9	61	48.4	53	47.3
test3 <i>slope</i>	62	59.0	64	59.0	63	59.9	65	59.5	63	60.3	65	59.7
test3 <i>step</i>	67	60.9	67	60.2	111	59.0	73	59.0	76	60.6	67	59.9
mcc1 <i>slope</i>	59	49.3	60	49.1	67	48.5	63	48.3	60	50.1	61	49.6
mcc1 <i>step</i>	83	51.3	69	49.6	123	48.5	81	48.5	89	51.5	68	49.9
mcc2 <i>slope</i>	295	216.0	291	211.7	363	186.3	344	186.0	303	215.6	293	210.9
mcc2 <i>step</i>	350	209.1	312	203.8	524	192.1	376	188.0	416	212.3	356	205.2

Table 2: Experiments on MCM benchmarks considering a variety of cost functions. We report tile-layer maximum and average congestion C_{max} and C_{avg} , and show the best and worst performance using a classic rip-up and reroute approach, and the results of applying a single pass of iterative deletion to the rip-up and reroute result combined with the two single-bend candidate paths. We also show results using a “typical” cost function configuration.

Benchmark	Single Layer		L	Multi Layer	
	Minotaur	V4R		Minotaur	V4R
test1	27	52	4	16	26
test2	50	78	4	29	34
test3	63	102	4	33	45
mcc1	60	94	4	48	74
mcc2	303	496	6	196	228

Table 3: Comparison of maximum tile-layer congestion of the MINOTAUR global routing result with that of V4R. For the single-layer comparison, the V4R layers were merged into a single layer.

4.3 Experiments with Physical Capacity Constraints

Table 4 shows experimental results and routings using a “two-tier” cost function and differing physical routing capacity constraints. Assuming a wire pitch of 75μ , we have a physical capacity per border of 36 for *mcc1* and 125 for *mcc2*. We also consider cases with increased wire pitch, resulting in physical capacities of 24, 20, 16, and 10 for *mcc1*, and 100, 80, and 60 for *mcc2*, respectively. We apply either an *ACE-slope* cost function, the “two-tier” function, or an *ACE-slope* approach followed by a post-processing step using the “two-tier” cost function. When physical constraints are relatively loose, using a two-tier cost function or *ACE-slope* followed by post-processing can further improve the average congestion (and expected wire length), as shown in the case with a routing capacity of 125 for *mcc2*. When an unrealistically tight capacity constraint is given (as is the case for the capacity of 60 for *mcc2*) the two-tier cost function may give a solution that is inferior to the one obtained using *ACE-slope*.

4.4 Performance Optimization

To evaluate the impact of performance optimization on global solution quality, we applied the *RATS-tree* algorithm to a number of nets from the benchmarks *mcc1* and *mcc2*. The number of nets optimized, minimum and maximum number of *RATS-trees*, and average number of *RATS-trees* per net are shown in Table 5. For these experiments, we constructed Steiner trees for each net, and evaluated the delay for each using *SPICE*. $T\%$ of the pins with highest delay were selected, and we attempt to obtain an $X\%$ reduction in signal delay for these pins (no delay constraints are placed on the other nets or pins). Router performance is shown in Table

<i>mcc1</i>				
	C_{max}	C_{avg}	Mono.	Non-Mono.
ACE-slope	17	11.4	1597	97
two-tier (35)	27	11.2	1678	16
two-tier (24)	21	11.2	1652	42
two-tier (20)	17	11.3	1622	72
two-tier (16)	16	11.7	1515	179
two-tier (10)	22	11.6	1535	159
post-process (35)	26	11.2	1668	26
post-process (24)	21	11.2	1645	49
post-process (20)	17	11.3	1630	64
post-process (16)	16	11.3	1625	69
post-process (10)	17	11.4	1597	97
<i>mcc2</i>				
	C_{max}	C_{avg}	Mono.	Non-Mono.
ACE-slope	84	48.7	6874	667
two-tier (125)	103	43.8	7412	129
two-tier (100)	84	44.5	6957	584
two-tier (80)	81	48.8	6191	1350
two-tier (60)	98	54.6	5406	2135
post-process (125)	103	43.7	7474	127
post-process (100)	83	44.5	7251	290
post-process (80)	80	48.8	6759	782
post-process (60)	84	48.7	6874	667

Table 4: Four-layer routing experiments on *mcc1* and *mcc2* with the border congestion metric, using physical routing capacity constraints (shown in parenthesis). We use either the *ACE-slope* cost function, a “two-tier” function, or the *ACE-slope* cost function followed by improvement with a “two-tier” function as a post-processing step. The number of monotonic and non-monotonic paths are also shown.

Bench mark	T	Total Nets	Min/max/avg. <i>RATS-trees</i>		
			$X = 5\%$	$X = 10\%$	$X = 15\%$
mcc1	5%	38	4/12/8.16	4/12/8.13	4/12/8.13
mcc1	10%	51	3/17/8.71	3/14/8.67	3/14/8.55
mcc2	5%	614	1/6/1.02	1/4/1.02	1/4/1.02
mcc2	10%	1278	1/6/1.01	1/4/1.01	1/4/1.01

Table 5: Number of nets with performance constraints, and the minimum, maximum, and average number of *RATS-trees* for each net.

Benchmark	T	Rip-up and reroute						Iterative deletion with RR postprocessing					
		$X = 5\%$		$X = 10\%$		$X = 15\%$		$X = 5\%$		$X = 10\%$		$X = 15\%$	
		C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}
mcc1	5%	34	22.6	34	22.6	34	22.6	32	22.6	33	22.7	33	22.7
mcc1	10%	41	22.3	41	22.3	41	22.3	33	22.8	33	22.8	33	22.8
mcc2	5%	184	93.6	184	93.6	184	93.6	184	93.6	184	93.6	185	93.6
mcc2	10%	223	93.3	223	93.3	223	93.3	222	93.3	222	93.3	222	93.3

Table 6: Performance optimization comparison of rip-up and reroute (using a single topology for critical nets) with iterative deletion followed by rip-up and reroute. In these experiments, we select $T\%$ of the high-delay pins, and attempt to obtain at least an $X\%$ delay reduction for each. In experiments with no performance optimization, rip-up and reroute obtains maximum and average congestion levels of 33 and 23.0 for mcc1, while congestion levels are 168 and 93.4 for mcc2. These results assume the single-layer routing model.

6. We consider two routing approaches for this problem. The first approach selects a single minimum-area candidate *RATS-tree* for each constrained net, and then applies rip-up and reroute the the remaining nets. The second approach is a hybrid, and applies iterative deletion to select a single *RATS-tree* for each constrained net, and then applies rip-up and reroute. When a number of candidate topologies are available, the hybrid approach obtains a 19.5% reduction in maximum congestion. In this set of experiments, the minimum-area *RATS-trees* were quite similar under the different performance constraints, resulting in similar performance for the rip-up and reroute approach.

5 Conclusions

We have presented an MCM/PCB global router which integrates the classic rip-up and reroute approach, with the modern iterative deletion method.

We draw a number of conclusions from our experiments. We find that slope-based cost functions are superior to step-based cost functions, rip-up and reroute performance is heavily dependent on cost function parameters, and net ordering has little impact for our routing model. Iterative deletion is effective for congestion minimization when we have signal delay and integrity performance constraints, or when rip-up and reroute cost function parameters are not well chosen.

We are currently integrating our MCM global router with a performance driven variable width multi-layer detail router, and will report results of these experiments when they are complete. We are also considering the use of a small set of candidate routes such as those suggested in [2], allowing us to eliminate maze routing entirely. Our methods may also extend to handle general area routing for performance driven multi-layer VLSI IC design problems.

6 Acknowledgements

This work is partially supported by DARPA/ETO under Contract DAAL01-96-K-3600 managed by the US Army Research Laboratory. The authors would like to thank Mr. Cheng-Kok Koh for his assistance and for supplying performance constraints and the *RATS-tree* algorithm.

References

- [1] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.
- [2] J. D. Carothers, D. Li, and T. Hameenanttila. MCG: A correct-by-design multichip module router with crosstalk avoidance. In *Int. Workshop on Rapid System Prototyping*, pages 183–188, 1996.
- [3] Y.-J. Cha, C. S. Rim, and K. Nakajima. A simple and effective greedy multilayer router for MCMs. In *Proc. Int. Symp. on Physical Design*, pages 67–72, 1997.
- [4] J. D. Cho, K.-F. Liao, S. Raje, and M. Sarrafzadeh. *M²R*: Multilayer routing algorithm for high-performance MCMs. *IEEE Trans. on Circuits and Systems*, 41(4):253–265, April 1994.
- [5] J. Cong, L. He, C.-K. Koh, and P. H. Madden. Performance optimization of VLSI interconnect layout. *Integration, the VLSI Journal*, 21:1–94, 1996.
- [6] J. Cong and C.-K. Koh. Interconnect layout optimization under higher-order RLC model. In *Proc. Int. Conf. on Computer Aided Design*, 1997.
- [7] J. Cong and P. H. Madden. Performance driven routing for standard cell design. In *Proc. Int. Symp. on Physical Design*, pages 73–80, 1997.
- [8] J. Cong and P. H. Madden. Performance driven multi-layer general area routing for pcb/mcm designs. Technical Report 980013, UCLA CS Dept, March 1998.
- [9] J. Cong and B. Preas. A new algorithm for standard cell global routing. In *Proc. Int. Conf. on Computer Aided Design*, pages 176–179, November 1988.
- [10] A. Hanafusa, Y. Yamashita, and M. Yasuda. Three-dimensional routing for multilayer ceramic printed circuit boards. In *Proc. Int. Conf. on Computer Aided Design*, pages 386–389, 1990.
- [11] A. B. Kahng and G. Robins. *On Optimal Interconnections for VLSI*. Kluwer Academic Publishers, 1994.
- [12] K. Kawamura, T. Shindo, T. Shibuya, H. Miwatari, and Y. Ohki. Touch and cross router. In *Proc. Int. Conf. on Computer Aided Design*, pages 56–59, 1990.
- [13] K.-Y. Khoo and J. Cong. A fast multilayer general area router for MCM designs. *IEEE Trans. on Circuits and Systems*, pages 841–851, 1992.
- [14] K.-Y. Khoo and J. Cong. An efficient multilayer MCM router based on four-via routing. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 1277–1290, 1994.
- [15] C. Y. Lee. An algorithm for path connection and its applications. *IRE Trans. on Electronic Computers*, EC-10(3):346–365, 1961.
- [16] R. Linsker. An iterative-improvement penalty-function-driven wire routing system. *IBM J. Research and Development*, 28(5):613–624, September 1984.
- [17] S. Mehrotra, P. Franzon, and M. Steer. Performance driven global routing and wiring rule generation for high speed PCBs and MCMs. In *Proc. Design Automation Conf*, pages 381–387, 1995.
- [18] H. Shin and A. Sangiovanni-Vincentelli. A detailed router based on incremental routing modifications: Mighty. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6(6):942–955, November 1987.
- [19] E. Shragowitz and S. Keel. A global router based on a multi-commodity flow model. *Integration, the VLSI Journal*, 5:3–16, 1987.
- [20] D. Wang and E. S. Kuh. A new timing-driven multilayer MCM/IC routing algorithm. In *Proc. IEEE Multi-Chip Module Conf.*, pages 89–94, 1997.
- [21] Q. Yu, S. Badida, and N. Sherwani. Algorithmic aspects of three dimensional MCM routing. In *Proc. Design Automation Conf*, pages 397–401, 1994.